Apostila do Curso de Criação de Filtros

Fábrica de Filtros @_fabricadefiltros

Curso de Criação de Filtros
com Meta Spark Studio e JavaScript
111 vídeos | 12h30m de conteúdo | 12 Projetos
Assista o Curso Básico Gratuito!
JS

Conheça mais em https://sites.google.com/view/meta-spark-studio

Índice Do Treinamento - 21 Dias - 12 Projetos Para Criar

Dia 01 - Curso Básico Gratuito - Primeiros Passos
Dia 02 - Projeto #01. Meta Studio Pro
Dia 03 - Projeto #02. Selfie com o Felpudo
Dia 04 - Projeto #03. Logotipos 3D
Dia 05 - Lógica 1. Variáveis e Operadores
Dia 06 - Lógica 2. Condição e Loops
Dia 07 - Lógica 3. Listas ou Arrays
Dia 08 - Lógica 4. Procedimentos Métodos e Funções
Dia 09 - Lógica 5. Método Async e Await e Gestures
Dia 10 - Exercício Path + Script 1 - Verificação da CNH
Dia 11 - Exercício Path + Script 2 - Jogo Par ou Ímpar
Dia 12 - Exercício Path + Script 3 - Jogo Jó Ken Pô (Pedra Papel ou Tesoura)
Dia 13 - Projeto #04. Frases e Pensamentos
Dia 14 - Projeto #05. Mobília / Escrivaninha 3D
Dia 15 - Projeto #06. Khan Academy 3D AR Portal
Dia 16 - Projeto #07. Personagem com Joystick Goku Dragon Ball-Z
Dia 17 - Projeto #08. Escultura com PolyScan Mestre Hsing Yün
Dia 18 - Projeto #09. Ciência de Dados Alimentação
Dia 19 - Projeto #10. Tamagotchi do Felpudo
Dia 20 - Projeto #11. Veículo Do Felpudo
Dia 21 - Projeto #12. AR Raposa Estelar

Dia 01 - Curso Básico Gratuito - Primeiros Passos

Download, Instalação e Primeiros Passos no Meta Spark Studio

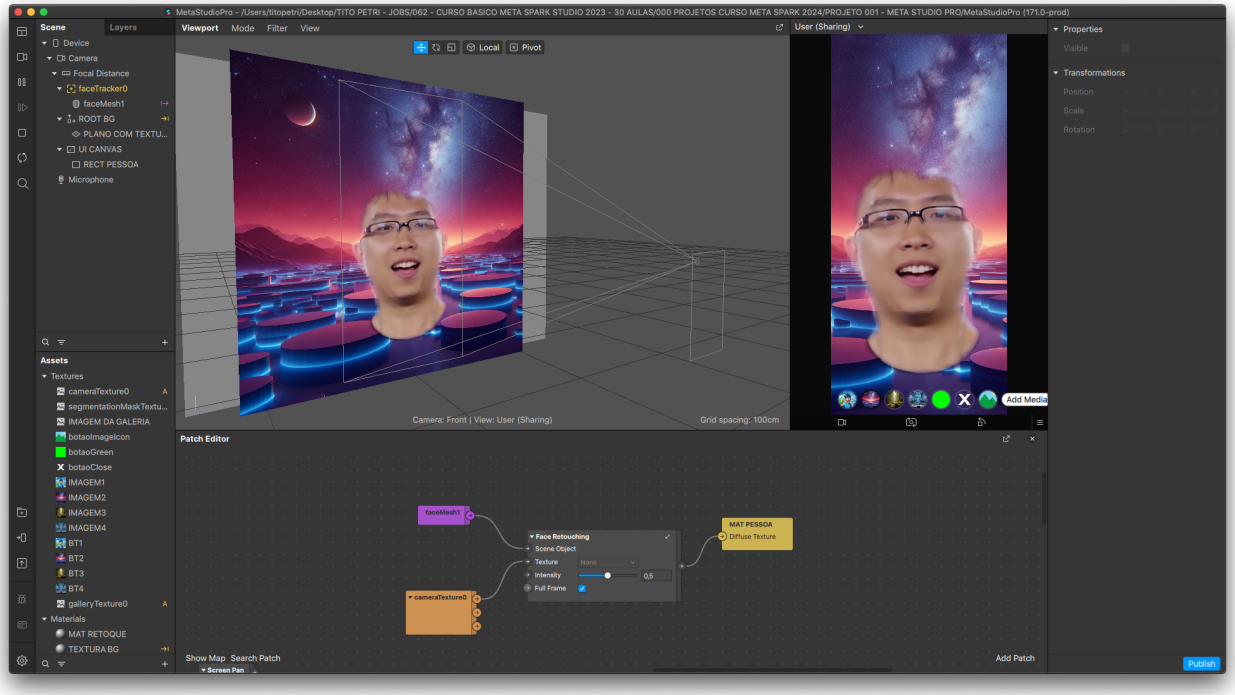
Curso Básico Gratuito de Meta Spark Studio
https://www.youtube.com/playlist?list=PLihNz8bQL2o70mgwssFw4wPkynJqYM-1y

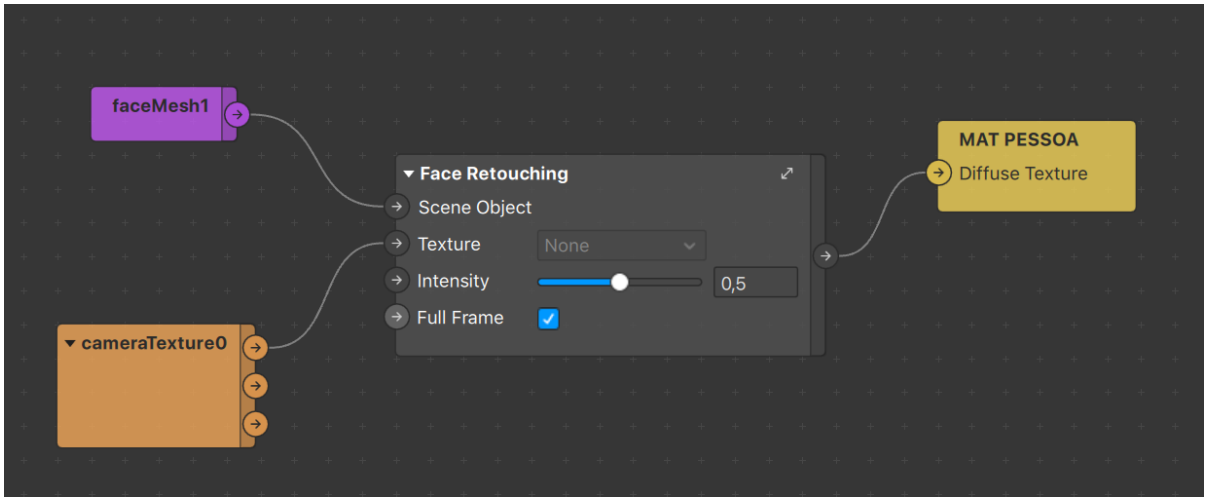
12 videos | 1:05 hrs.

1. Introdução ao Meta Spark Studio
2. Requisitos Mínimos para Desenvolver
3. Links Úteis para Download
4. Diferentes Simuladores do Spark AR
5. Principai Ferramentas Comandos Menus e Atalhos
6. Testes no Aparelho e Publicação do Filtro
7. Hierarquia de Transformações - Global e Local
8. Editor de Patches
9. Ajustando Texturas com 2D Texture Pack
10. Gradientes, Transições e Tipos
11. Meu Primeiro Script
12. Biblioteca de Realidade Aumentada (AR Library) e Sketchfab

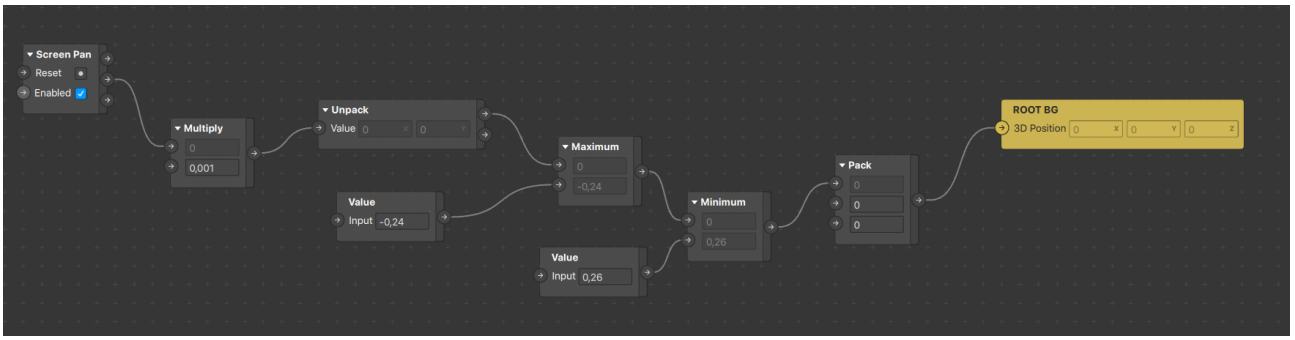
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

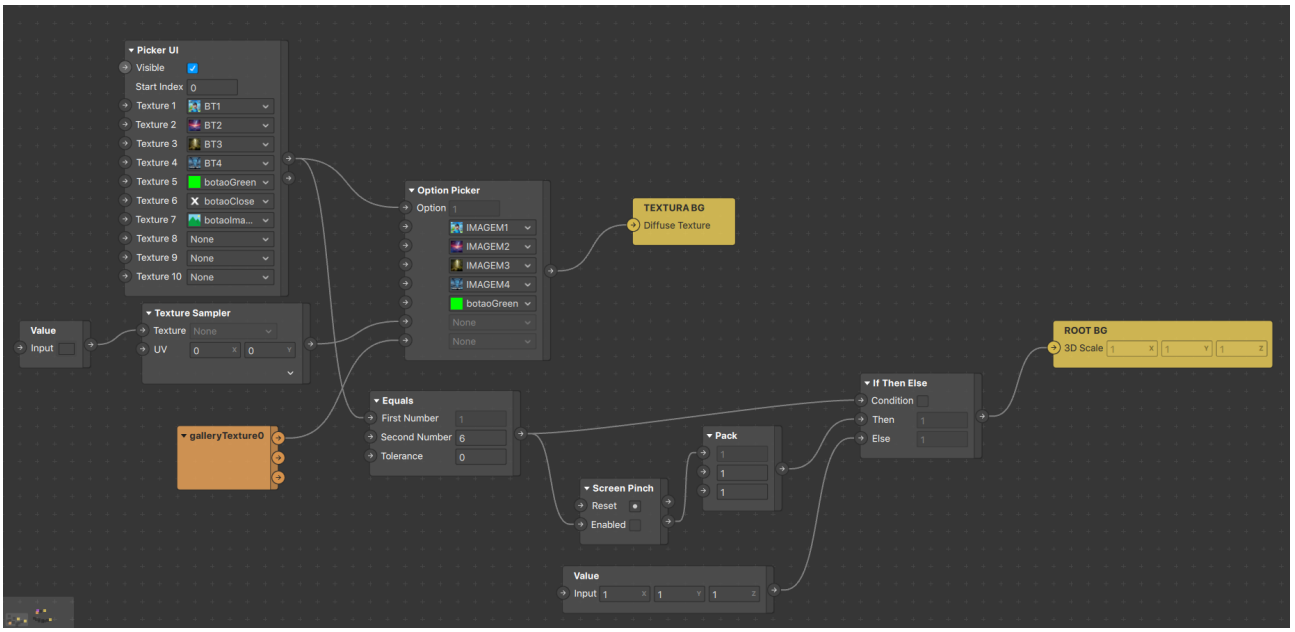
Dia 02 - Projeto #01. Meta Studio Pro





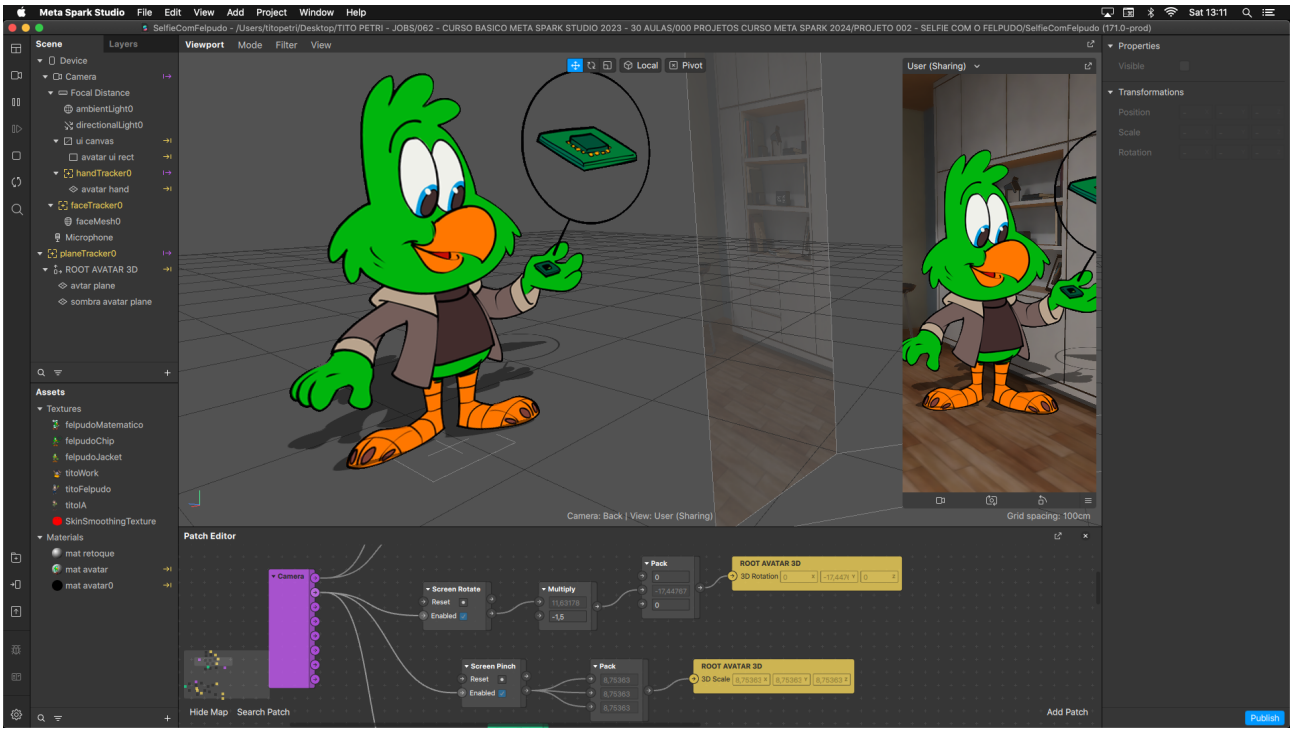
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

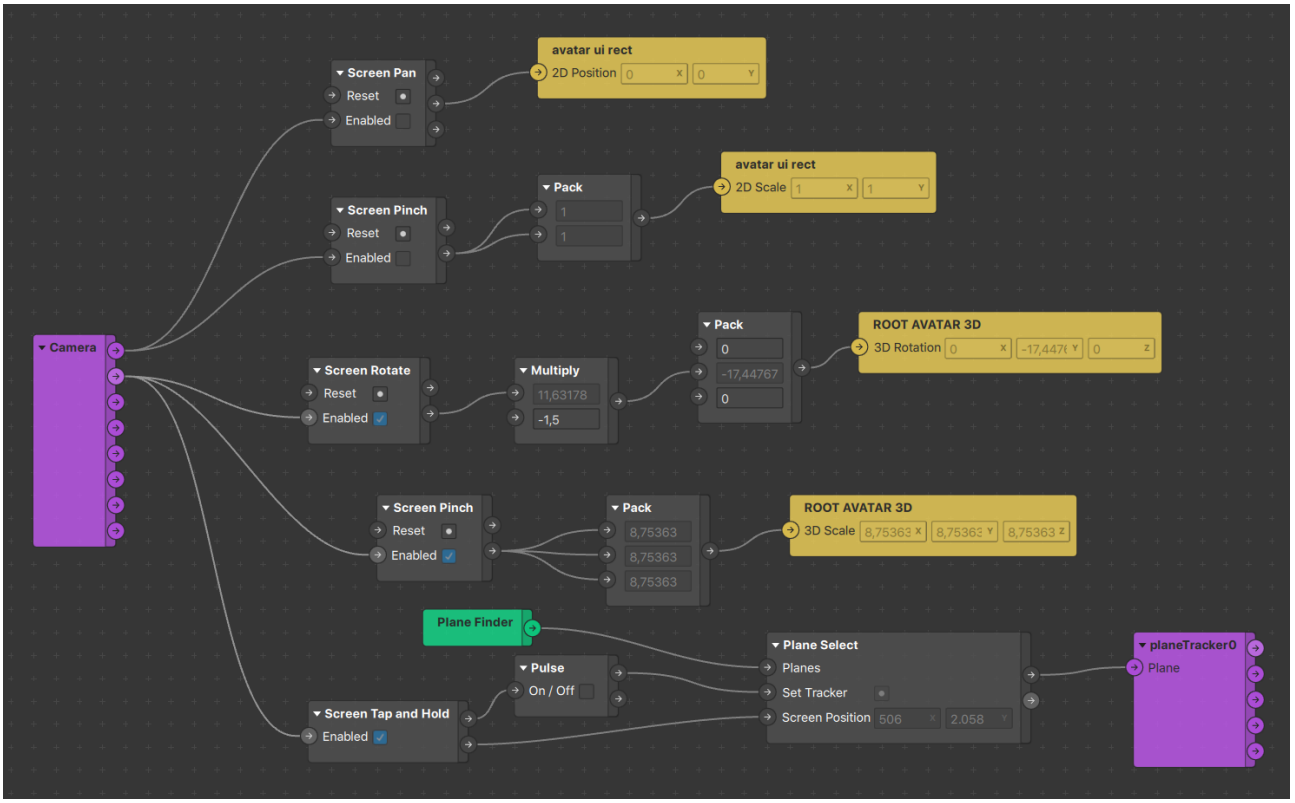




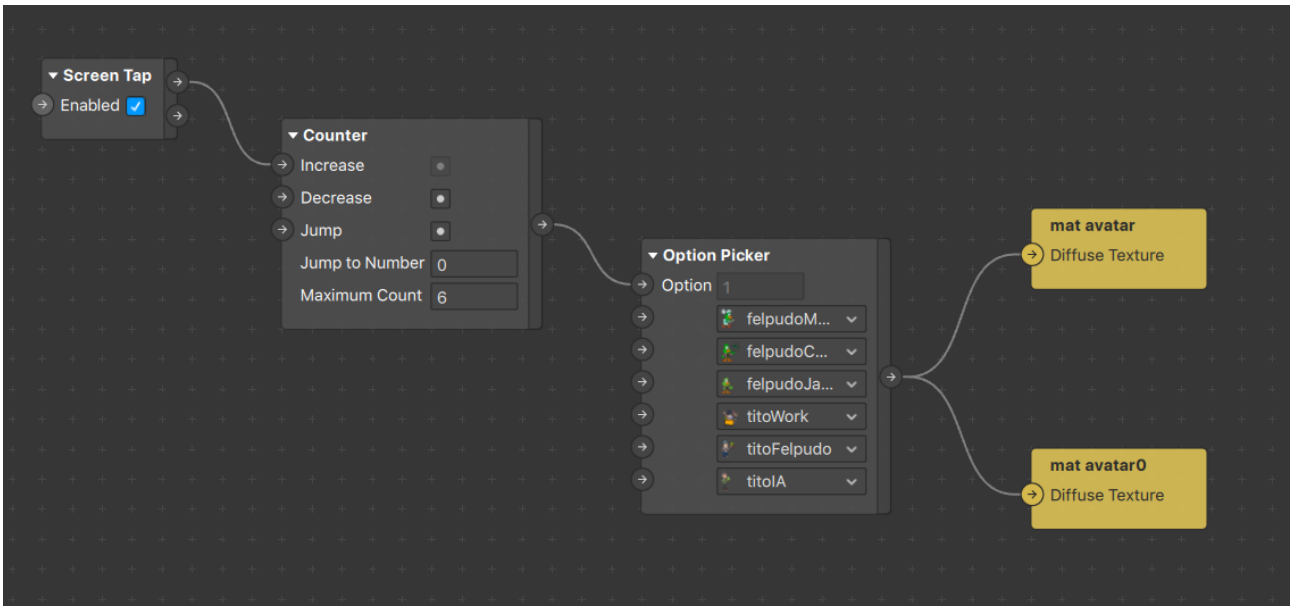
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

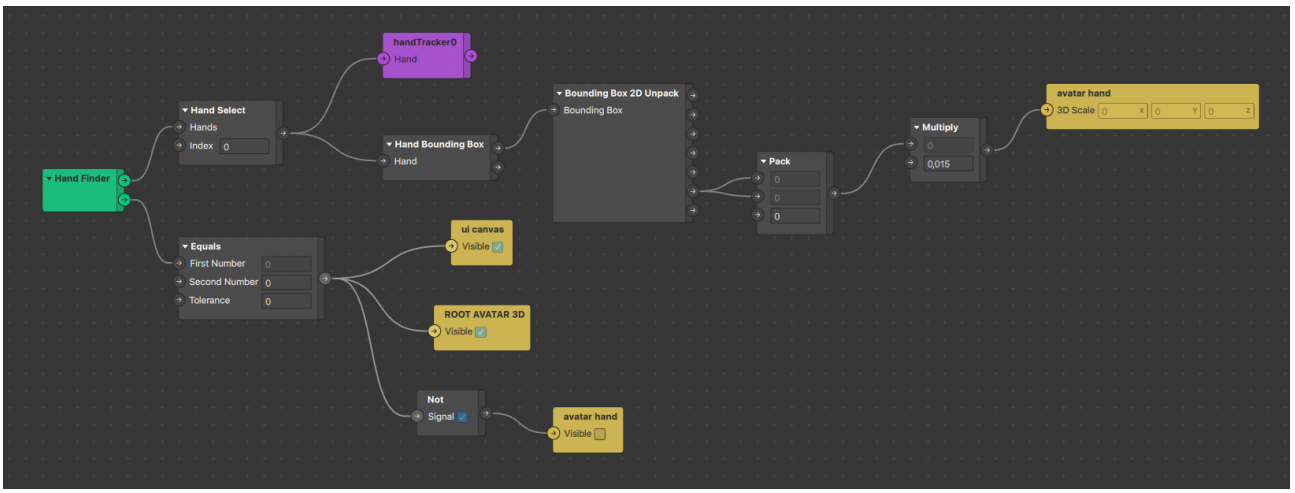
Dia 03 - Projeto #02. Selfie com o Felpudo





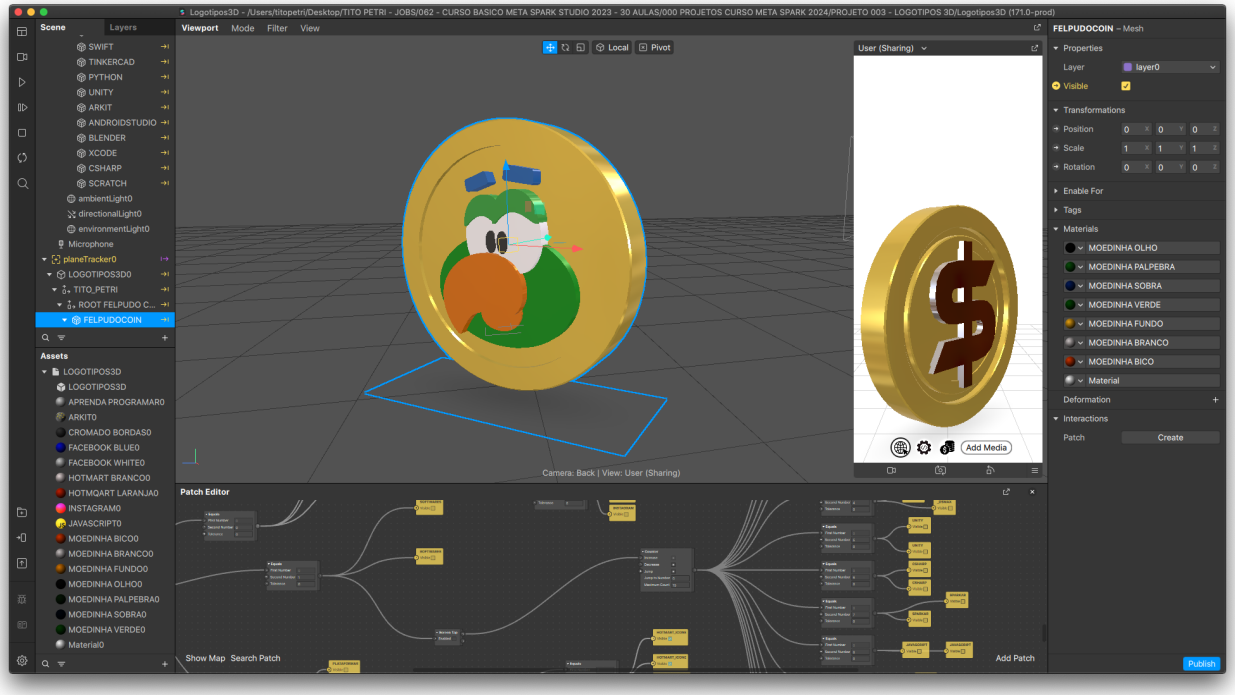
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

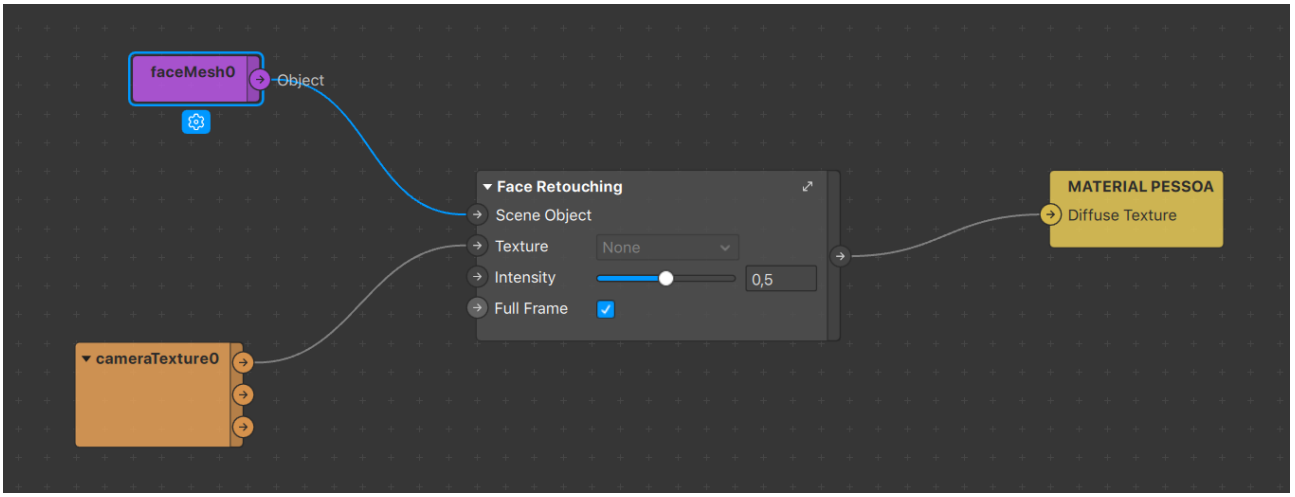


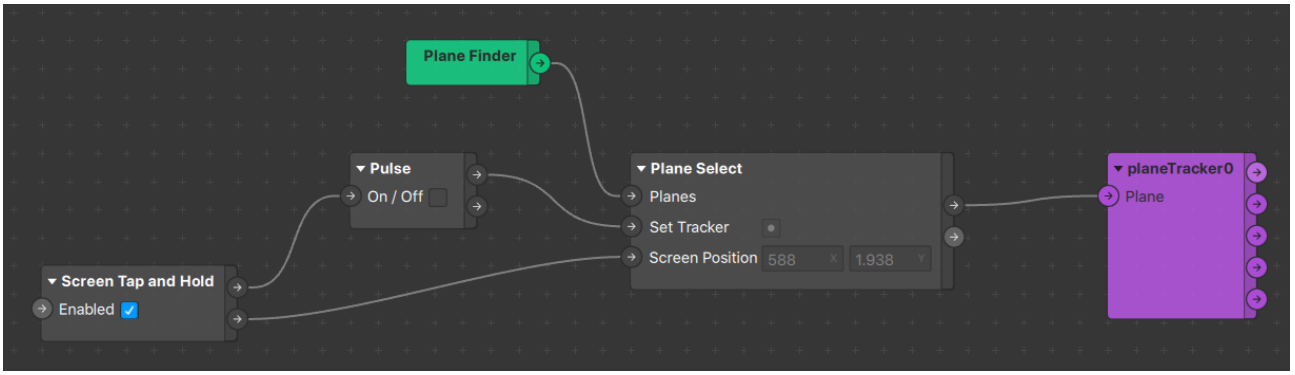


©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

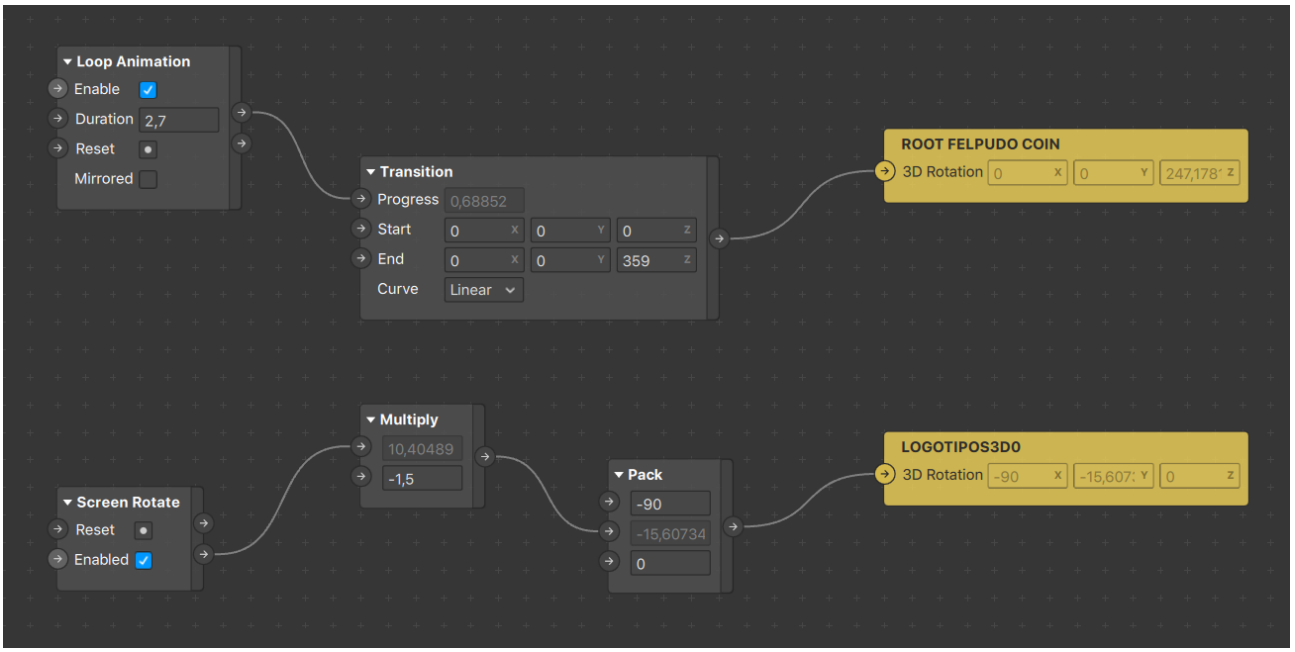
Dia 04 - Projeto #03. Logotipos 3D

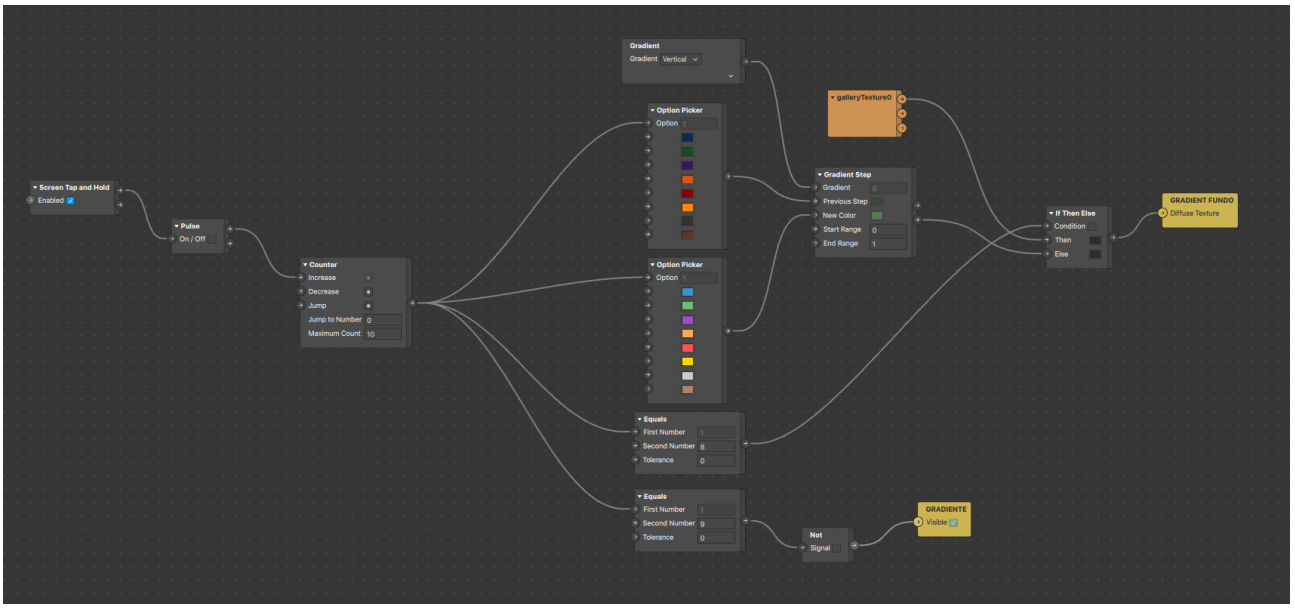


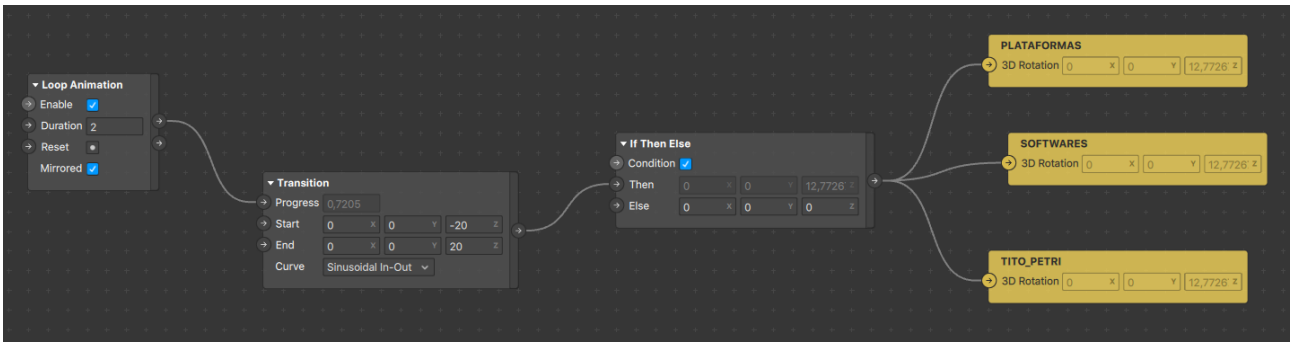




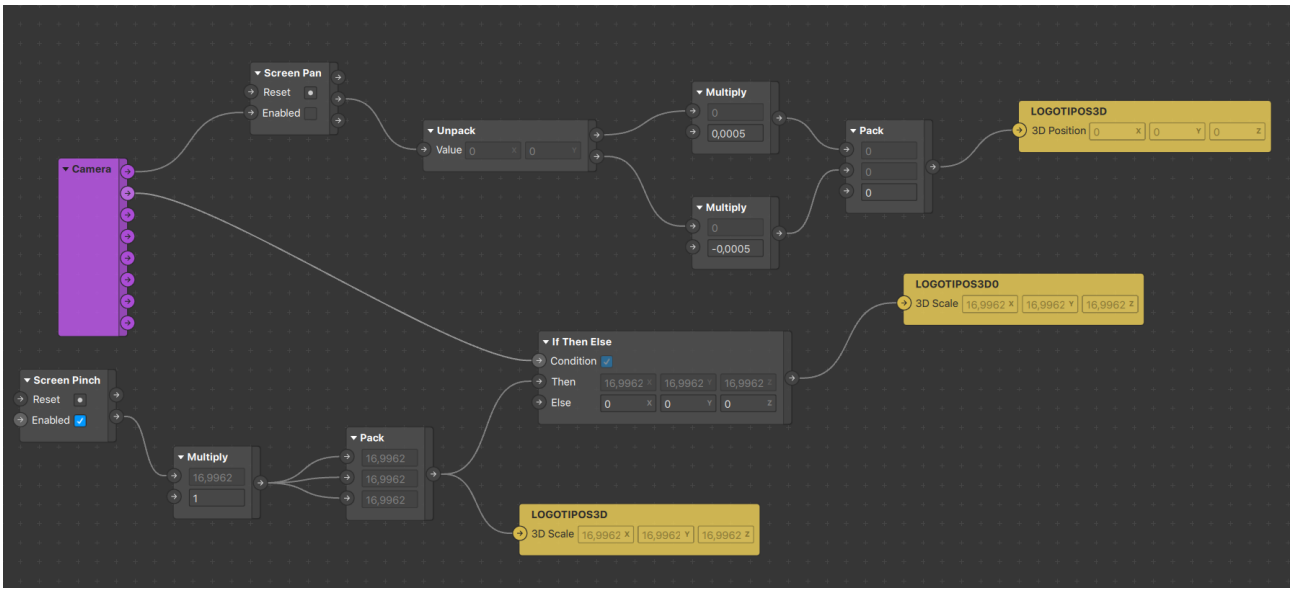
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

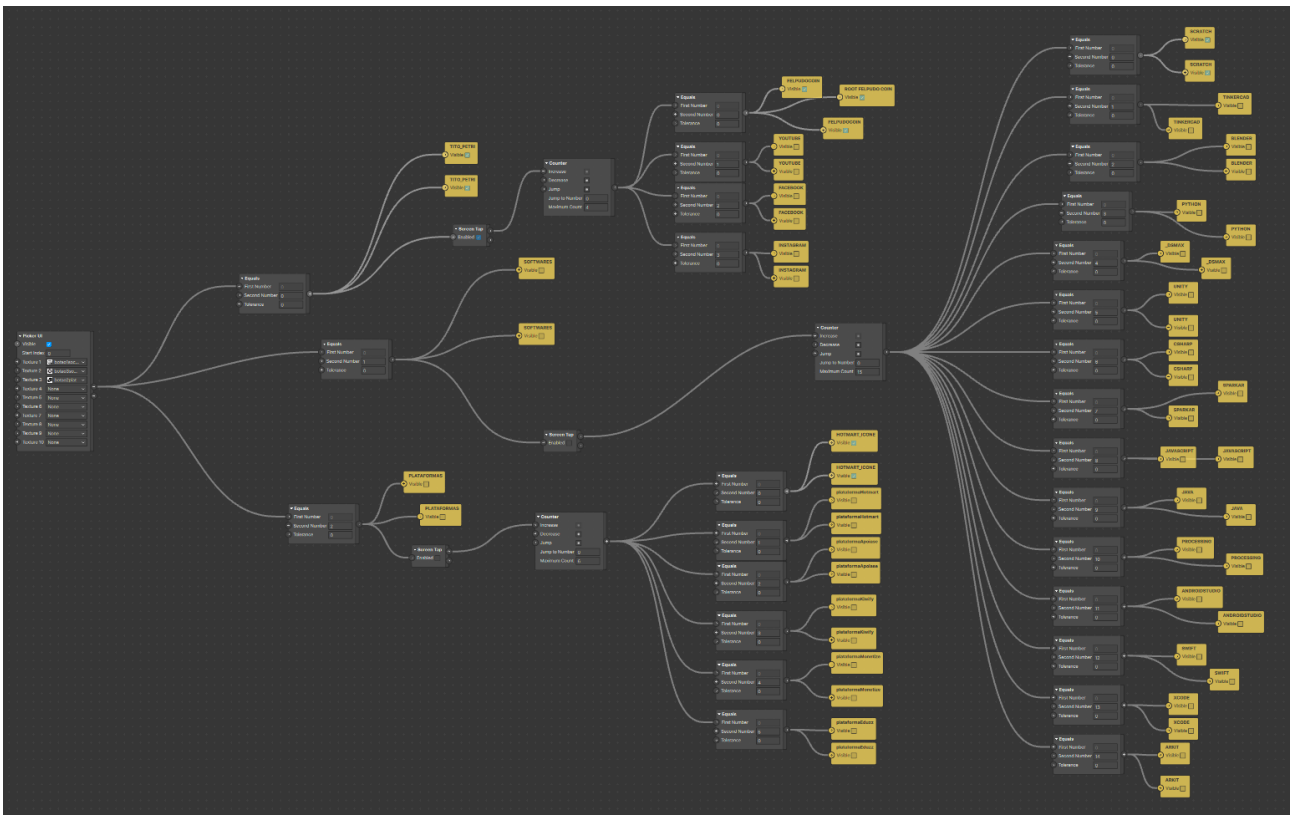






©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri





©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

Dia 05 - Lógica 1. Variáveis e Operadores

```
// carregamento dos módulos
const Reactive = require('Reactive');
const Blocks = require('Blocks');
const Scene = require('Scene');
const Diagnostics = require('Diagnostics');
const TouchGestures = require('TouchGestures');
const Time = require('Time');
const Patches = require('Patches');
const Materials = require('Materials');
const Textures = require('Textures');
const NativeUI = require('NativeUI');
const Animation = require('Animation');
const FaceTracking = require('FaceTracking');
const CameraInfo = require('CameraInfo');

// variáveis
let nome = "Tito Petri";
nome = "Felpudinho";

// constantes valores fixos
const meuNumero = 10;
const meuNome = "Felpudo";
const pi = 3.14159;
const opcao = true;

const meuValor = null;
const meuValor2 = undefined;

// operações aritméticas
Diagnostics.log(10+10);
Diagnostics.log(10-10);
Diagnostics.log(10*10);
Diagnostics.log(10/10);
Diagnostics.log(10%3);
Diagnostics.log(10**3);

// operadores unários
let contador = 10;
contador = contador + 1;
contador++;
contador = contador + 5;
contador += 5;
contador %= 10;

// operadores lógicos
Diagnostics.log(true);
Diagnostics.log(!true); // not
Diagnostics.log(true && true && false); // e and
Diagnostics.log(true || false || false || false ); // ou or

// operadores de comparação
Diagnostics.log(10 == 10);
Diagnostics.log(10 != 10);
Diagnostics.log(10 < 10);
Diagnostics.log(10 <= 10);
Diagnostics.log(10 > 5);
Diagnostics.log(10 >= 5);
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

```
/*
 Em JavaScript, o operador === é chamado de
 "operador de igualdade estrita"
 ou "operador de igualdade rigorosa".
 Ele é usado para comparar dois valores em JavaScript
 para verificar se eles são iguais tanto em valor
 quanto em tipo de dados.
*/

Diagnostics.log(true === true);
Diagnostics.log(null === undefined);
Diagnostics.log(5 === 15);

1 === 1 // true, pois ambos os valores são do tipo número e têm o mesmo valor
1 === '1' // false, pois um é um número e o outro é uma string
1 == '1' // true, pois o operador de igualdade solta converte '1' para número antes da comparação

Diagnostics.log("✅ Bem-Vindo ao Meta Spark Studio & JavaScript ✨");
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

Dia 06 - Lógica 2. Condição e Loops

```
const Diagnostics = require('Diagnostics');

// escopos ou limites
const minhaVariavel = 0;

if(true){
 Diagnostics.log(minhaVariavel);
}

// estruturas de condição
 if(false){
   Diagnostics.log("verdade");
 }else{
   Diagnostics.log("mentira");
 }

const idade = 20;

if(idade < 18){
 Diagnostics.log("Não Pode tirar CNH");
}else if(idade > 60){
 Diagnostics.log("Deve Renovar a Carta");
}else{
 Diagnostics.log("Pode tirar CNH");
}

let meuNome = "Fofura";

switch(meuNome){
 case "Felpudo":
   Diagnostics.log("Oi Felpudo!");
   Diagnostics.log("Seja Bem Vindo");
 break;

 case "Tito": Diagnostics.log("Oi Programador!"); break;
 default: Diagnostics.log("Nome Desconhecido!"); break;
}

// loops ou laços de repetição FOR
// 3 args - INICIALIZAÇÃO DO CONTADOR | CONDIÇÃO | INCREMENTO

for(let i=10; i>=0; i--){
 Diagnostics.log("Contador i: " + i);
}

let meuContador = 0;

while(meuContador<5){
 Diagnostics.log(meuContador);
 meuContador++;
}
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

Dia 07 - Lógica 3. Listas ou Arrays

```
const Diagnostics = require('Diagnostics');

// LISTAS OU ARRAYS
let numero = 10;
let numeros = [10,20,30,100,500];

const personagens = ["Felpudo", "Fofura", "Lesmo", "Buguinho", "Uruca"];
Diagnostics.log(personagens[0]);
Diagnostics.log(personagens.length);
Diagnostics.log(personagens[personagens.length-1]);

Diagnostics.log(personagens.shift());
Diagnostics.log(personagens.push("Peluche", "Tito Jr."));
Diagnostics.log(personagens.unshift("Felpudinho", "Aprenda Programar"));

Diagnostics.log(personagens);
Diagnostics.log(personagens.sort());

const listaNumeros = [7, 2, 5, 1, 9, 3];
Diagnostics.log(listaNumeros.sort());

const listaNumeros3 = [1, 2, 3, 4, 5];
listaNumeros3.splice(1, 2);
Diagnostics.log(listaNumeros3);

for(let i=0; i<personagens.length; i++){
 Diagnostics.log(personagens[i]);
}

personagens.forEach(nome => {
 Diagnostics.log("Nome: " + nome);
});

Diagnostics.log("✅ Bem-Vindo ao Meta Spark Studio & JavaScript ✨");
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

Dia 08 - Lógica 4. Procedimentos Métodos e Funções

```
const Diagnostics = require('Diagnostics');
const minhasFrases = ["Frase 1", "Frase 2", "Frase 3"];
Diagnostics.log(minhasFrases[0]);

const dicionarioFrases = [
 {autor:"Walt Disney", frase:"Se você pode sonhar, você pode fazer."},
 {autor:"Seu Madruga", frase:"A vingança nunca é plena, mata a alma e envenena."},
 {autor:"Chaves", frase:"Prefiro morrer do que perder a vida"},
 {autor:"Kiko", frase:"Espera até eu ganhar minha bola quadrada."},
];

Diagnostics.log(dicionarioFrases[0].autor);
Diagnostics.log(dicionarioFrases[0].frase);

for(let i=0;i<dicionarioFrases.length;i++){
 Diagnostics.log("Frase: " + dicionarioFrases[i].frase + "\nAutor: " + dicionarioFrases[i].autor);
}

// MÉTODOS (PROCEDIMENTOS E FUNÇÕES)

let i = 0;
mostraMensagemOK();
imprima("Tito Petri");
const resultado = somaValores(10,5);
imprima(resultado);

function mostraMensagemOK(){
 Diagnostics.log(i);
 Diagnostics.log("✅ Bem-Vindo ao Meta Spark Studio & JavaScript ✨");
}

function imprima(mensagem){
 Diagnostics.log(mensagem);
}

function somaValores(a,b){
 return a+b;
}
```

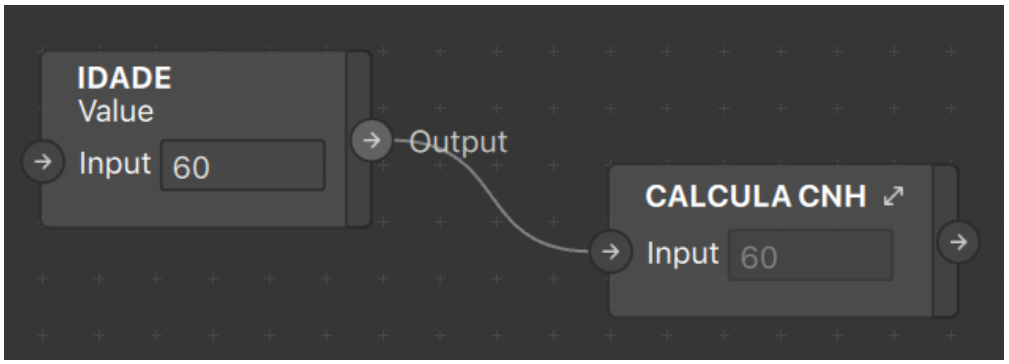
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

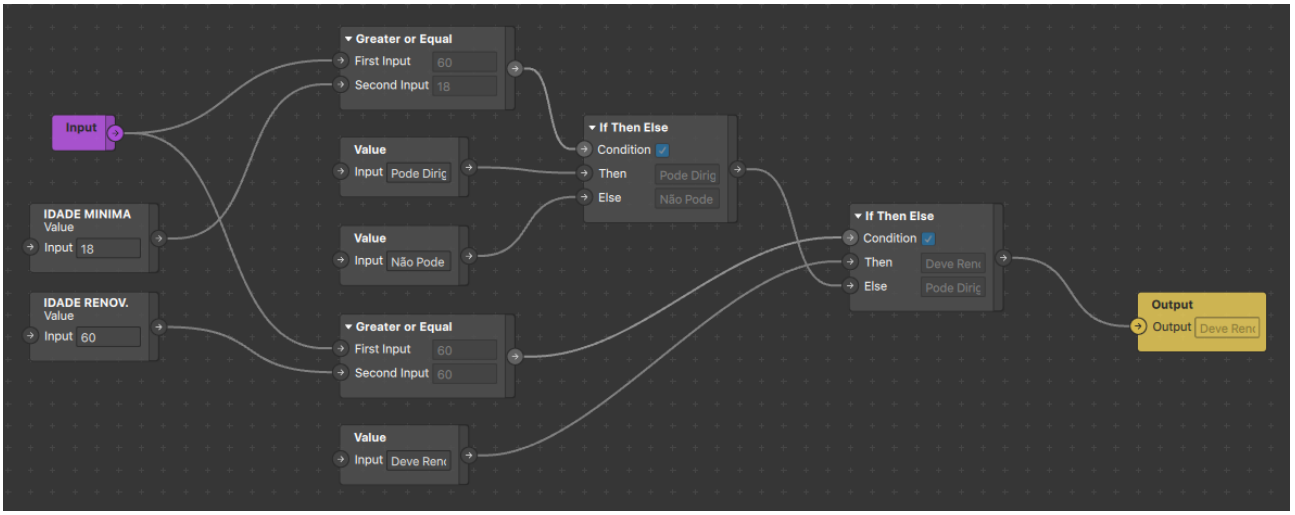
Dia 09 - Lógica 5. Método Async e Await e Gestures

```
//const Diagnostics = require('Diagnostics');

import { Diagnostics, funcaoTesteX } from './meuNovoScript.js'

const Scene = require('Scene');
const TouchGestures = require('TouchGestures');

(async function(){

 const focalNode = await Scene.root.findFirst("Focal Distance");
 const meuTexto2D = await Scene.root.findFirst("meuTexto");

 const [ambientLight, directionalLight] = await Promise.all([
   Scene.root.findFirst('ambientLight0'),
   Scene.root.findFirst('directionalLight0'),
 ]);
  meuTexto2D.text = "Felpudinho!";

 Diagnostics.watch("Intensidade Luz Ambiente: ",ambientLight.intensity);
 Diagnostics.watch("Intensidade Luz: ",directionalLight.intensity);

 TouchGestures.onTap().subscribe((gesture) => {
   Diagnostics.log("Tap Screen");
   meuTexto2D.text = "Aula 05 de JavaScript!!!";
 });

})();

funcaoTesteX();
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

Dia 10 - Exercício Path + Script 1 - Verificação da CNH

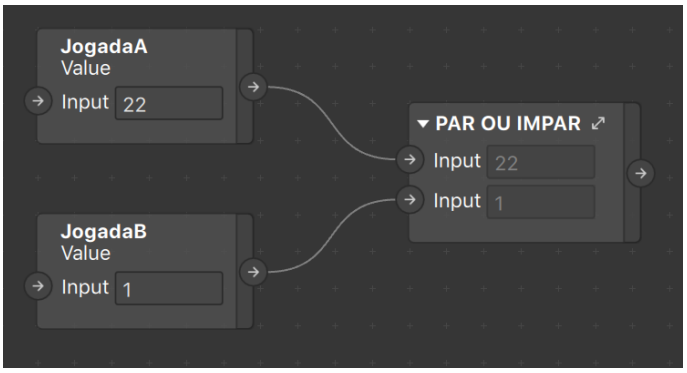


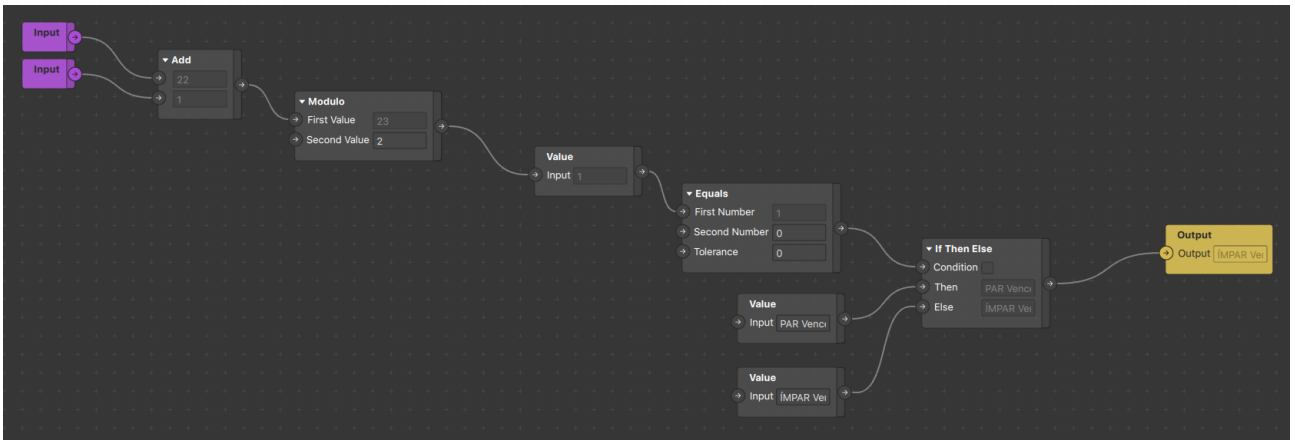



```
const Scene = require('Scene');
export const Diagnostics = require('Diagnostics');

let idade = 61;
const idadeMinima = 18;
const idadeRenovacao = 60;

(async function(){
   const meuTexto = await Scene.root.findFirst('3dText0');
    if(meuTexto != null){
     calculaCNH();
     meuTexto.text = calculaCNH();
   }
})();

function calculaCNH(){
 let msgResposta = "";
 if(idade >= idadeRenovacao){
   msgResposta = "Deve renovar a Carteira.";
 }else if(idade >= idadeMinima){
   msgResposta = "Pode Dirigir.";
 }else{
   msgResposta = "Não pode Dirigir.";
 }
 return msgResposta;
}

// Diagnostics.log("✨ Script Compilado com Sucesso! ✅");
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

Dia 11 - Exercício Path + Script 2 - Jogo Par ou Ímpar

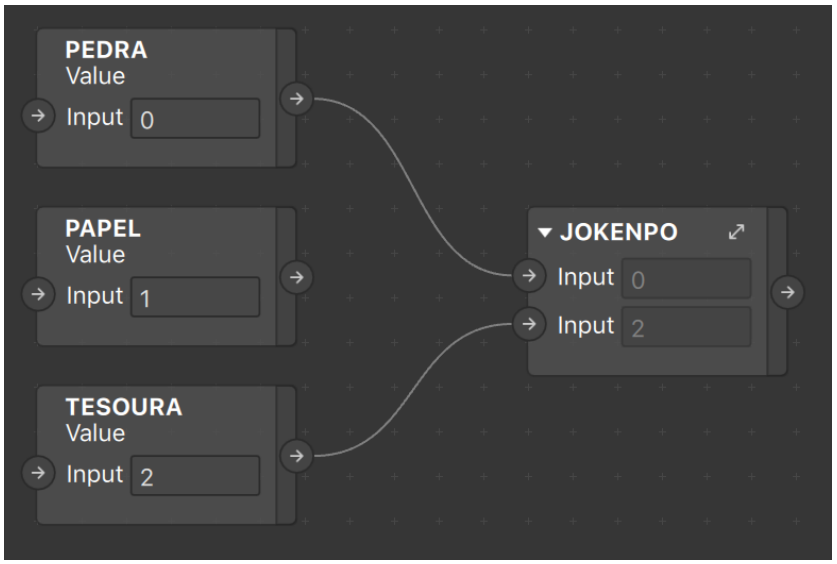


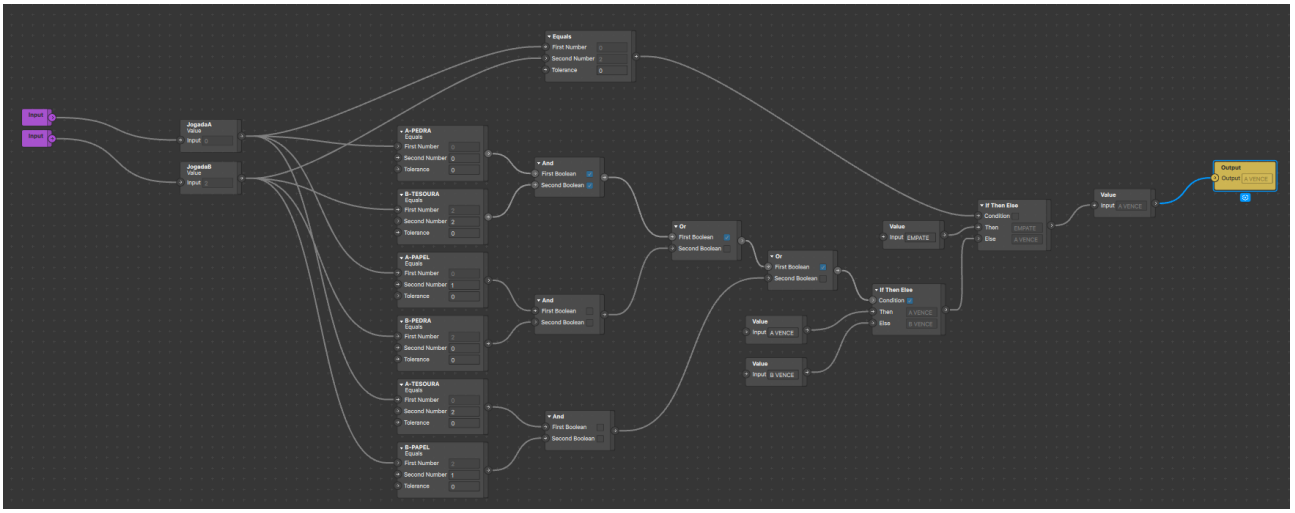



```
export const Diagnostics = require('Diagnostics');
const Scene = require('Scene');

let jogadaA = Math.floor(Math.random() * 5) + 1;;
let jogadaB = Math.floor(Math.random() * 5) + 1;;

//verificaResultado();

(async function(){
  const texto = await Scene.root.findFirst('3dText0');
  if(texto!=null){
    texto.text = verificaResultado();
  }
})();

function verificaResultado(){
  Diagnostics.log("A:"+jogadaA);
  Diagnostics.log("B:"+jogadaB);
  let resultado = "";
  if((jogadaA+jogadaB)%2==0){
    resultado = "PAR Venceu!";
  }else{
    resultado = "ÍMPAR Venceu!";
  }
  return resultado;
}

Diagnostics.log("✨ Script Compilado com Sucesso! ✅");
```

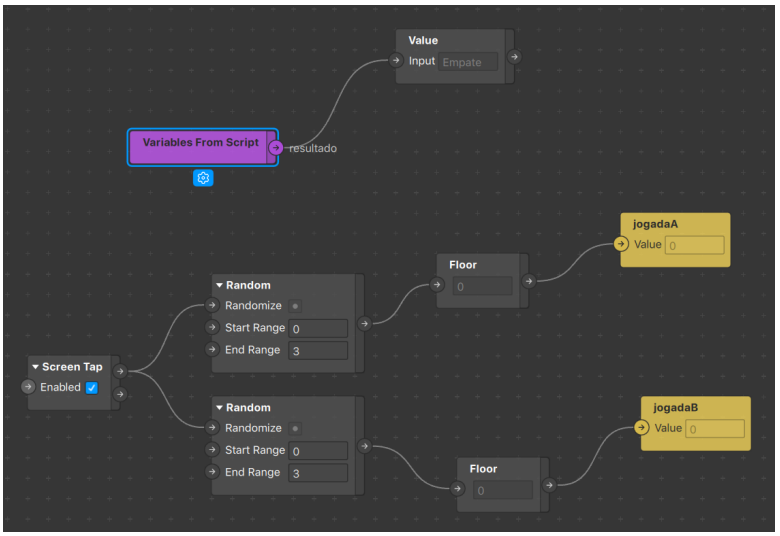
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

Dia 12 - Exercício Path + Script 3 - Jogo Jó Ken Pô (Pedra Papel ou Tesoura)





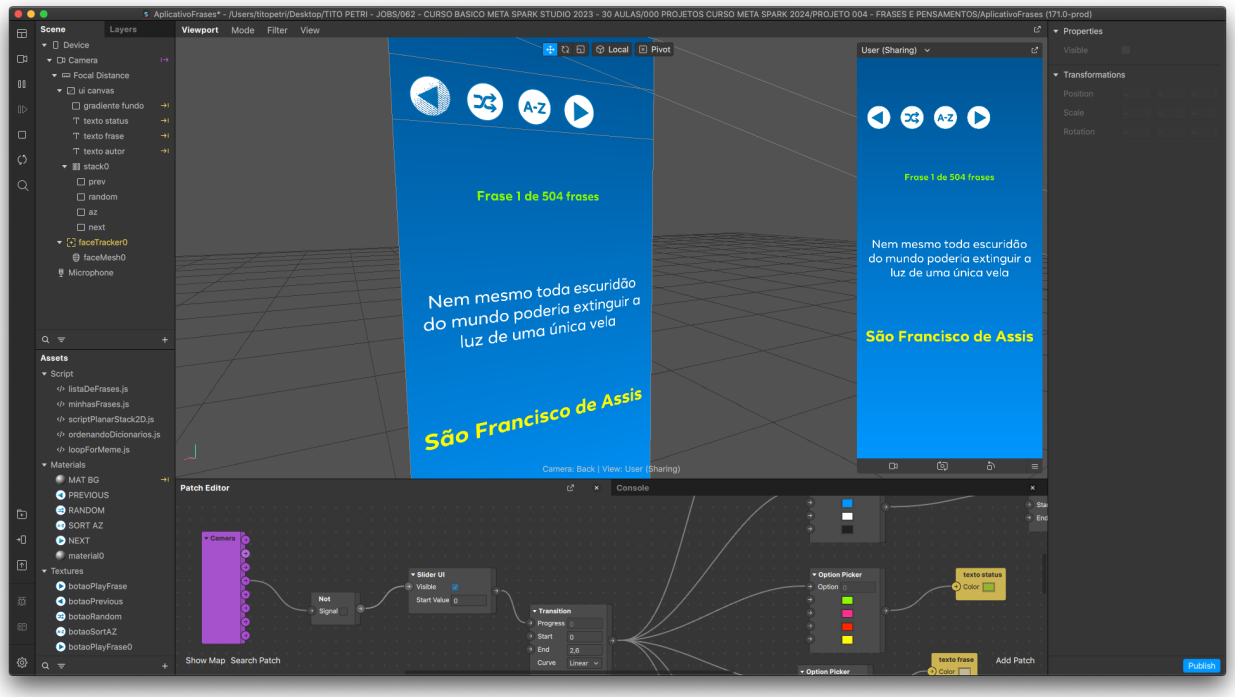
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

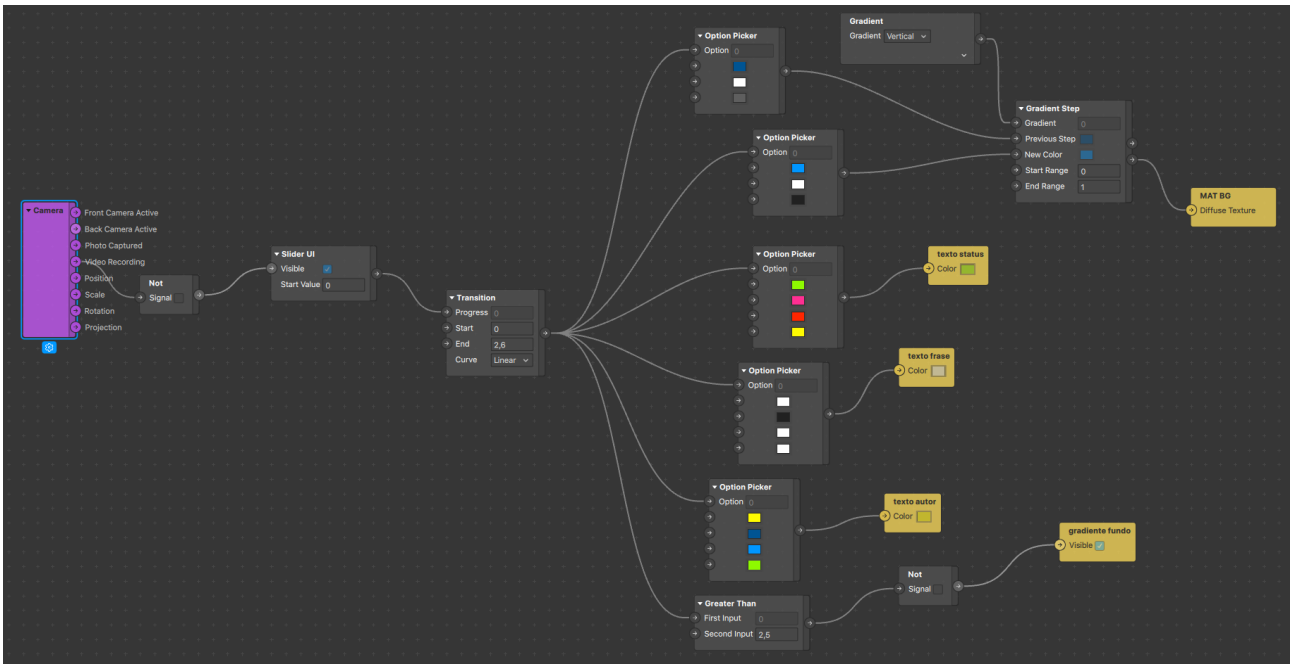



```
const Diagnostics = require('Diagnostics');
const Patches = require('Patches');

let jogadaA;
let jogadaB;

let resultado = "";
let emojis = ["👊","🖐️","✌️"];
let palavras = ["pedra","papel","tesoura"];

( async function(){
  await Patches.outputs.getScalar('jogadaA').then(event => {
     event.monitor({ fireOnInitialValue: true }).subscribe(function (valores) {
        jogadaA=valores.newValue;
        confereResultado();
     });
  });

  await Patches.outputs.getScalar('jogadaB').then(event => {
    event.monitor({ fireOnInitialValue: true }).subscribe(function (valores) {
       jogadaB=valores.newValue;
       confereResultado();
    });
  });
})();

function confereResultado(){
 if(jogadaA != null && jogadaB != null){
  if(jogadaA==jogadaB){
    resultado = "Empate";
  }else if((jogadaA==0 && jogadaB==2)||(jogadaA==1 && jogadaB==0)||(jogadaA==2 && jogadaB==1)){
    resultado = "Jogador A Venceu";
  } else{
    resultado = "Jogador B Venceu";
  }

  Diagnostics.log("Jogada A: " + emojis[jogadaA] +" | "+ palavras[jogadaA]);
  Diagnostics.log("Jogada B: " + emojis[jogadaB] +" | "+ palavras[jogadaB]);
  Diagnostics.log("\n" + resultado);
  Patches.inputs.setString('resultado', resultado);
 }
}
Diagnostics.log("✨ Script Compilado com Sucesso! ✅");
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

Dia 13 - Projeto #04. Frases e Pensamentos


Frase 1 de 504 frases
Nem mesmo toda escuridão do mundo poderia extinguir a luz de uma única vela
São Francisco de Assis
Patch Editor
Console



©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

```
const Scene = require('Scene');
const Diagnostics = require('Diagnostics');
const TouchGestures = require('TouchGestures');
const Time = require('Time');

const CameraInfo = require('CameraInfo');

import { listaDeFrases } from './listaDeFrases.js'
import { ordenaListaAZ } from './ordenandoDicionarios.js'

let indiceFrase = 0;
let intervalTimer = Time.setInterval(atualizaTexto,25);

let texto = "";
let contador = 0;
let textoFrase;
let textoAutor;

function atualizaTexto(){
  if(contador<texto.length){
    textoFrase.text = texto.substring(0,contador);
    contador++;
  }else{
    stopTimer();
  }
}
let isRecordingVideo;

function stopTimer(){
  if(textoAutor!=null){
    textoAutor.hidden = false;
  }
  Time.clearInterval(intervalTimer);
}

function startTimer(){
  contador = 0;
  intervalTimer = Time.setInterval(atualizaTexto,25);
}

( async function () {
    isRecordingVideo = CameraInfo.isRecordingVideo;
    isRecordingVideo.onOn({fireOnInitialValue: true}).subscribe((event) => {
      Diagnostics.log("Start Recording...");
      atualizarUITextos();

    });
    textoFrase = await Scene.root.findFirst('texto frase');
    textoAutor = await Scene.root.findFirst('texto autor');

    const [textoStatus, btPrev, btRandom, btNext, btAz] = await Promise.all([
      //Scene.root.findFirst('texto frase'),
      //Scene.root.findFirst('texto autor'),
      Scene.root.findFirst('texto status'),
      Scene.root.findFirst('prev'),
      Scene.root.findFirst('random'),
      Scene.root.findFirst('next'),
      Scene.root.findFirst('az')
    ]);


    TouchGestures.onTap(btPrev).subscribe(() => {
      Diagnostics.log("Clicou Botão Prev");
      if(indiceFrase>0){indiceFrase--;}
      atualizarUITextos();
    });
    TouchGestures.onTap(btRandom).subscribe(() => {
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

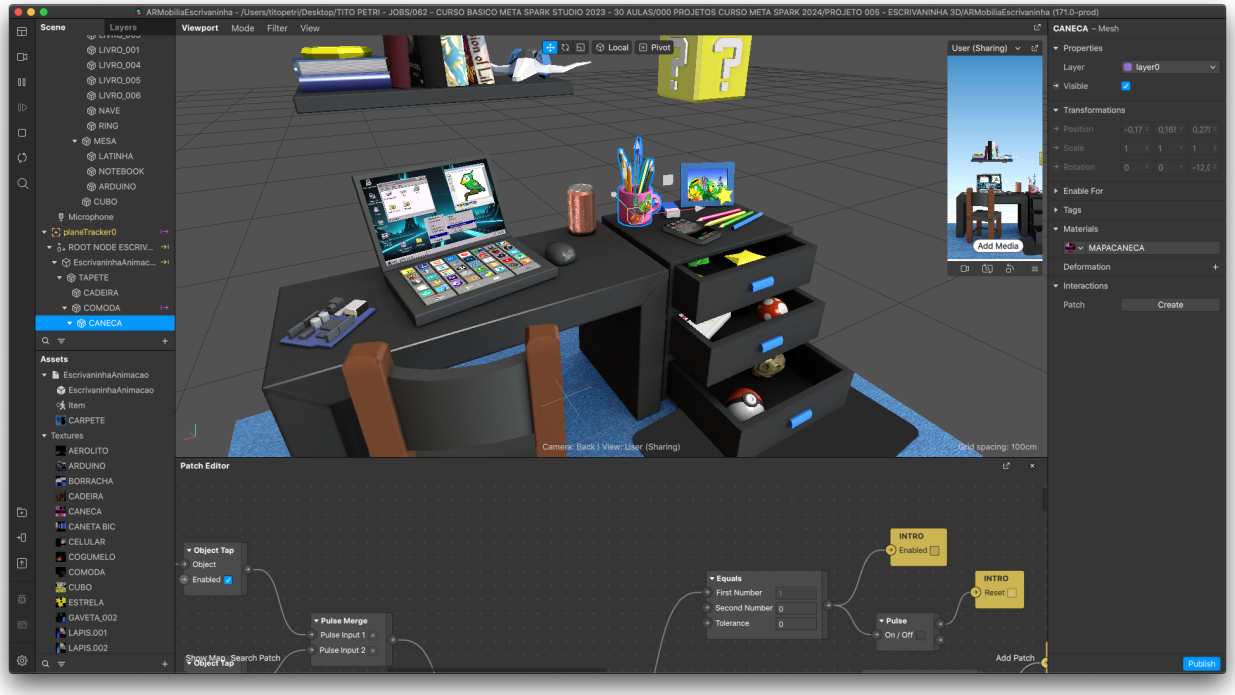
```
    Diagnostics.log("Clicou Botão Random");
    randomizarIndice();
  });
  TouchGestures.onTap(btNext).subscribe(() => {
    Diagnostics.log("Clicou Botão Next");
    if(indiceFrase<listaDeFrases.length-1){indiceFrase++;}
    atualizarUITextos();
  });
  TouchGestures.onTap(btAz).subscribe(() => {
    Diagnostics.log("Clicou Botão A-Z");
    ordenaListaAZ();
    indiceFrase = 0;
    atualizarUITextos();
  });

  function atualizarUITextos() {
    texto = listaDeFrases[indiceFrase].frase;
    textoFrase.text = listaDeFrases[indiceFrase].frase;
    if(listaDeFrases[indiceFrase].autor=="{"){
      textoAutor.text = "Autor Desconhecido"
    }else{
      textoAutor.text = listaDeFrases[indiceFrase].autor;
    }
    textoAutor.hidden = true;
    textoStatus.text = "Frase " +(indiceFrase+1).toString()+" de "+listaDeFrases.length.toString() + " frases";
    textoFrase.text=" ";
    startTimer();
  }

  function randomizarIndice(){
    const min = 0;
    const max = listaDeFrases.length-1;
    const randomInteger = Math.floor(Math.random() * (max - min + 1)) + min;
    indiceFrase = randomInteger;
    atualizarUITextos();
  }

  atualizarUITextos();


})();
Diagnostics.log("✅ Compilado com Sucesso!✨");


const Diagnostics = require("Diagnostics");
import { listaDeFrases } from './listaDeFrases.js'

export function ordenaListaAZ(){

// Ordenar primeiro por autor e depois por pensamento

listaDeFrases.sort((a, b) => {
  // Primeiro, comparar por autor
  const autorA = a.autor.toLowerCase();
  const autorB = b.autor.toLowerCase();
  if (autorA < autorB) return -1;
  if (autorA > autorB) return 1;

  // Se os autores forem iguais, comparar por pensamento

  const pensamentoA = a.frase.toLowerCase();
  const pensamentoB = b.frase.toLowerCase();
  if (pensamentoA < pensamentoB) return -1;
  if (pensamentoA > pensamentoB) return 1;

  return 0;
});
```

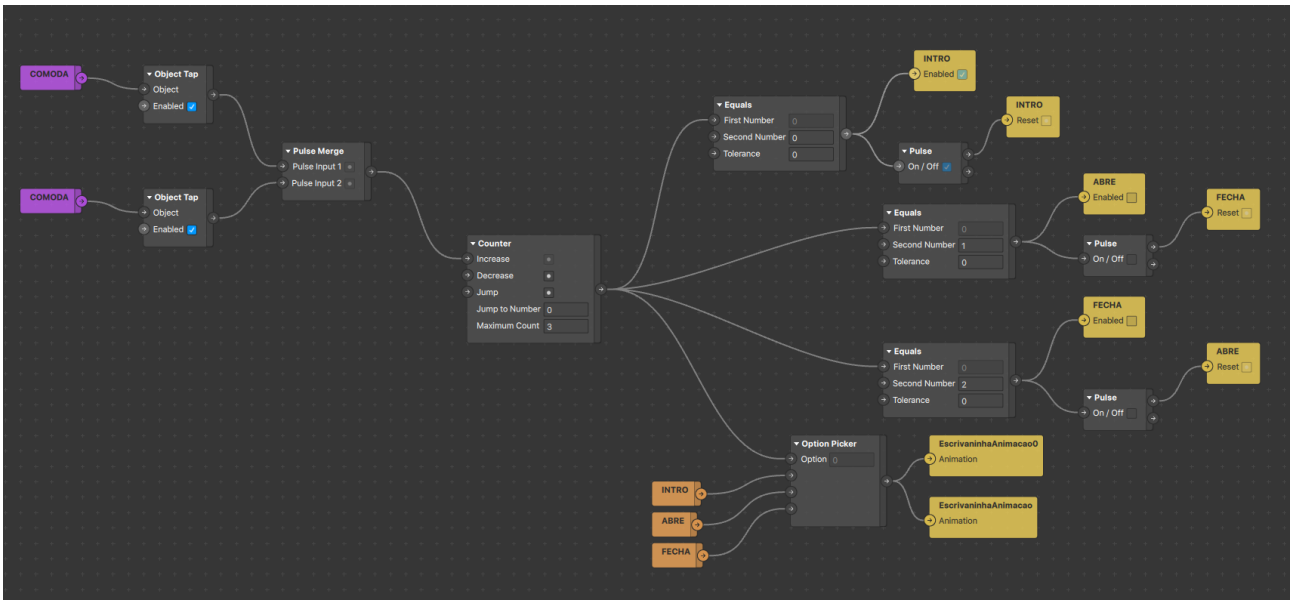
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

```
Diagnostics.log(listaDeFrases);
}

// array frases

export const listaDeFrases = [
    { frase: "Nem mesmo toda escuridão do mundo poderia extinguir a luz de uma única vela.", autor: "São Francisco de Assis"},
    { frase: "Não podemos resolver problemas usando o mesmo tipo de pensamento que usamos quando os criamos.", autor: "Albert Einstein"},
    { frase: "Sua dor é a quebra da concha que aprisiona sua compreensão.", autor: "Kahlil Gibran"},
    { frase: "Há duas maneiras de espalhar a luz: ser a vela ou o espelho que a reflete."}
];
```

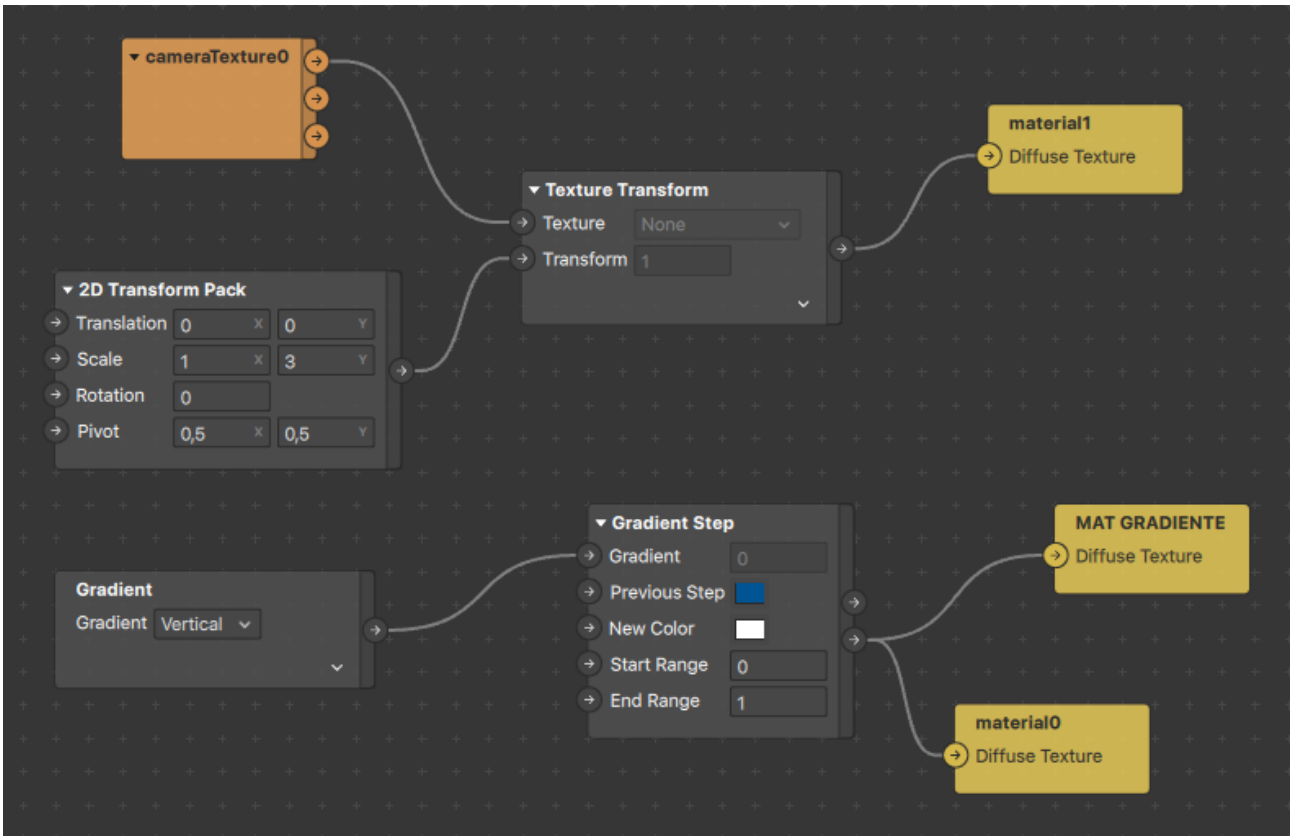
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

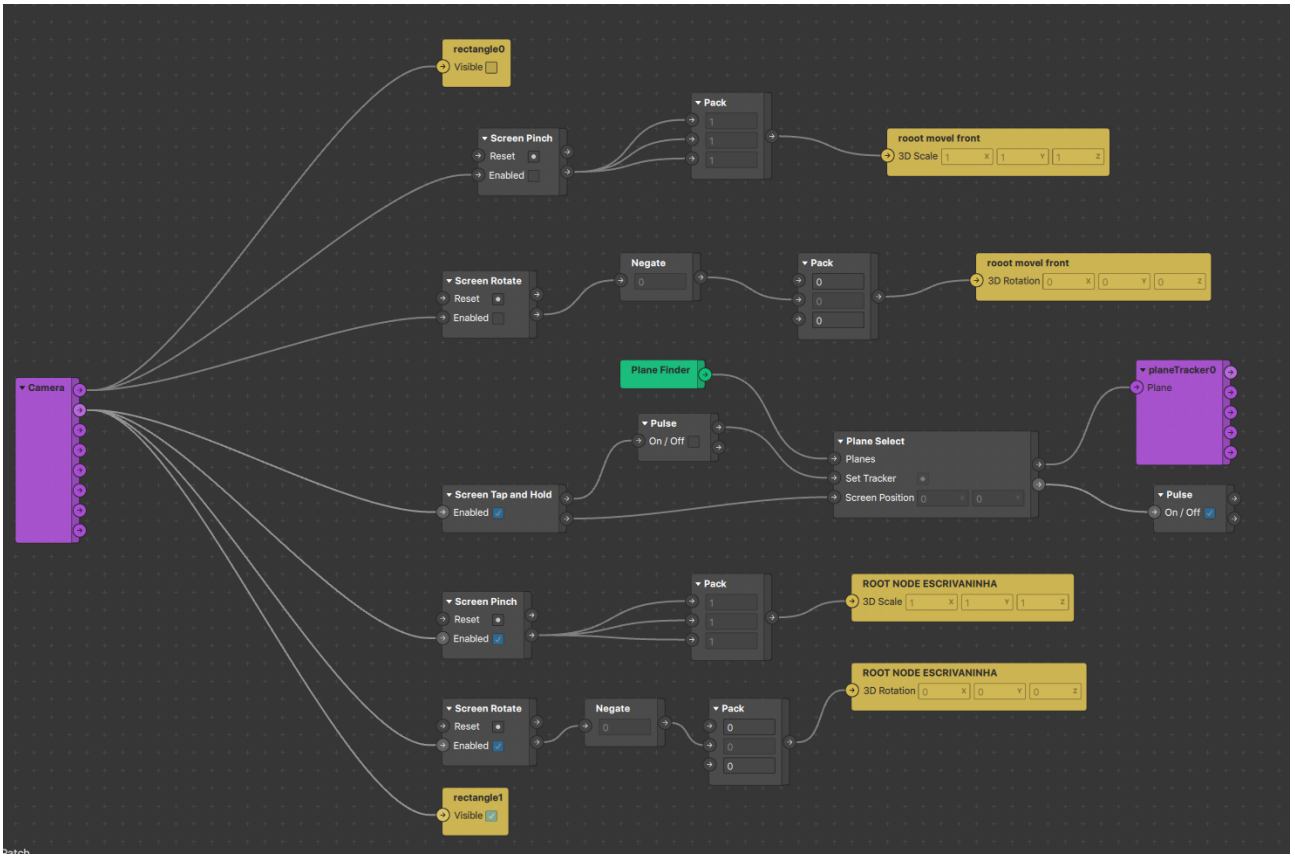
Dia 14 - Projeto #05. Mobília / Escrivaninha 3D





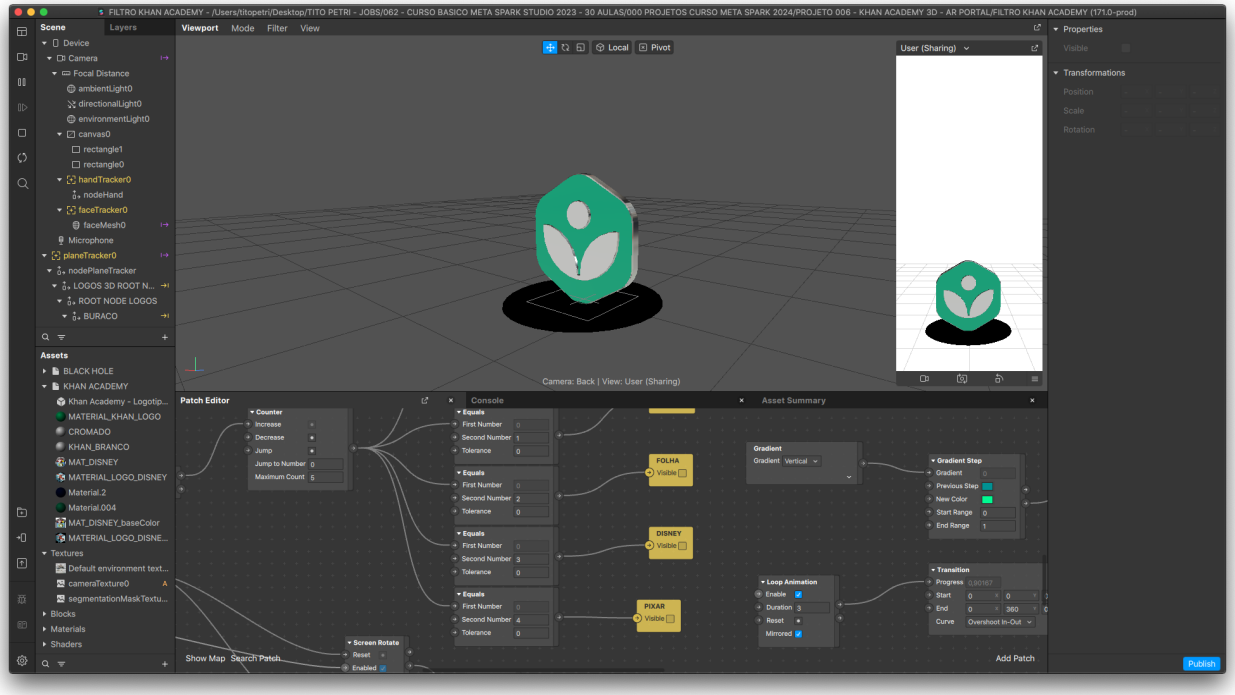
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

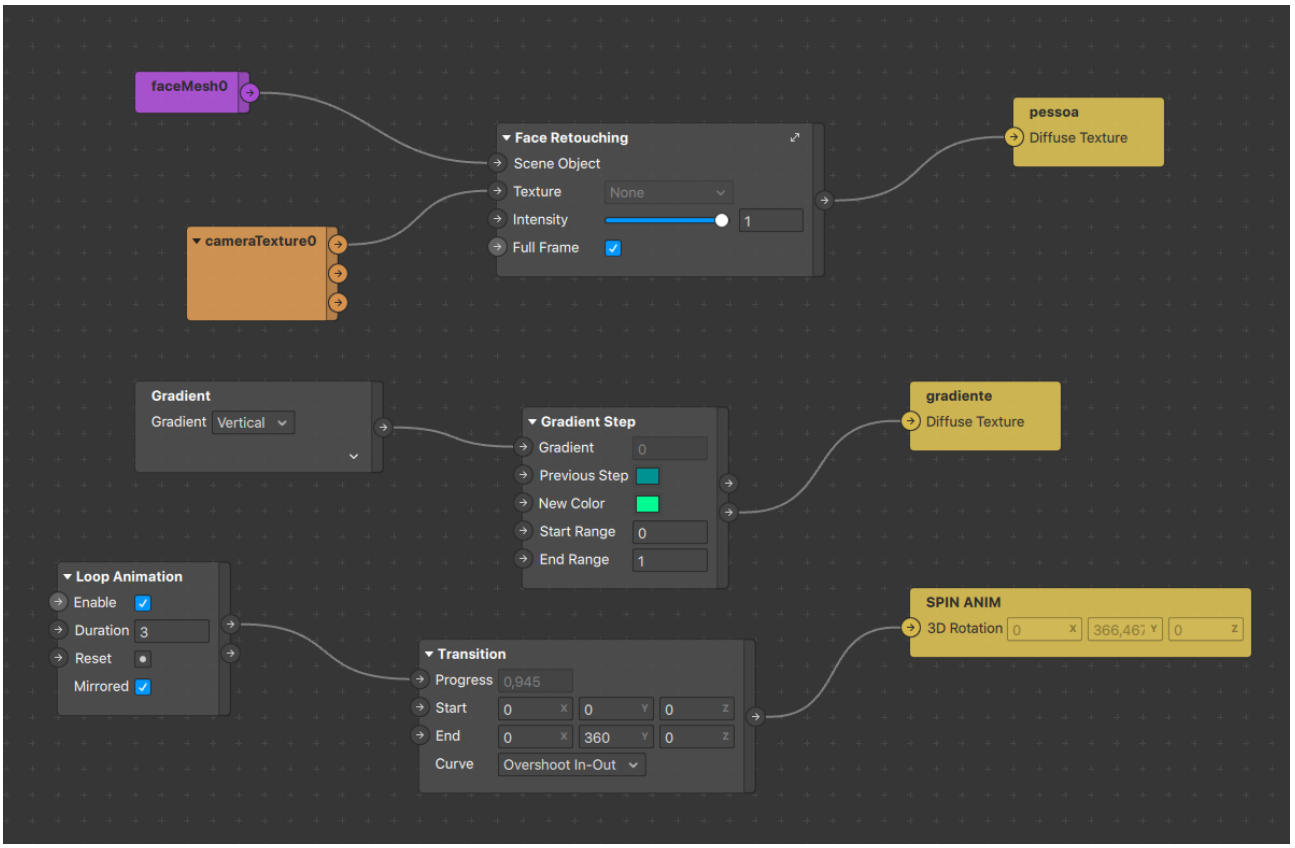




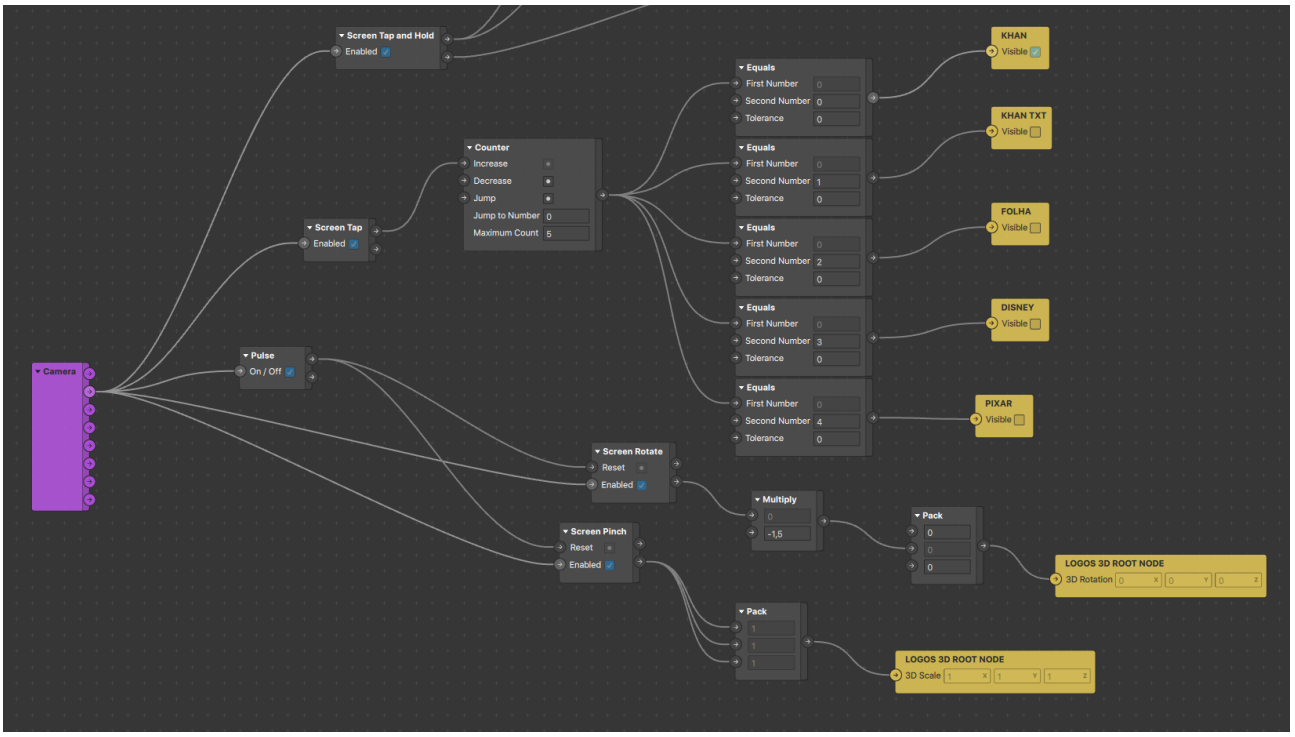
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

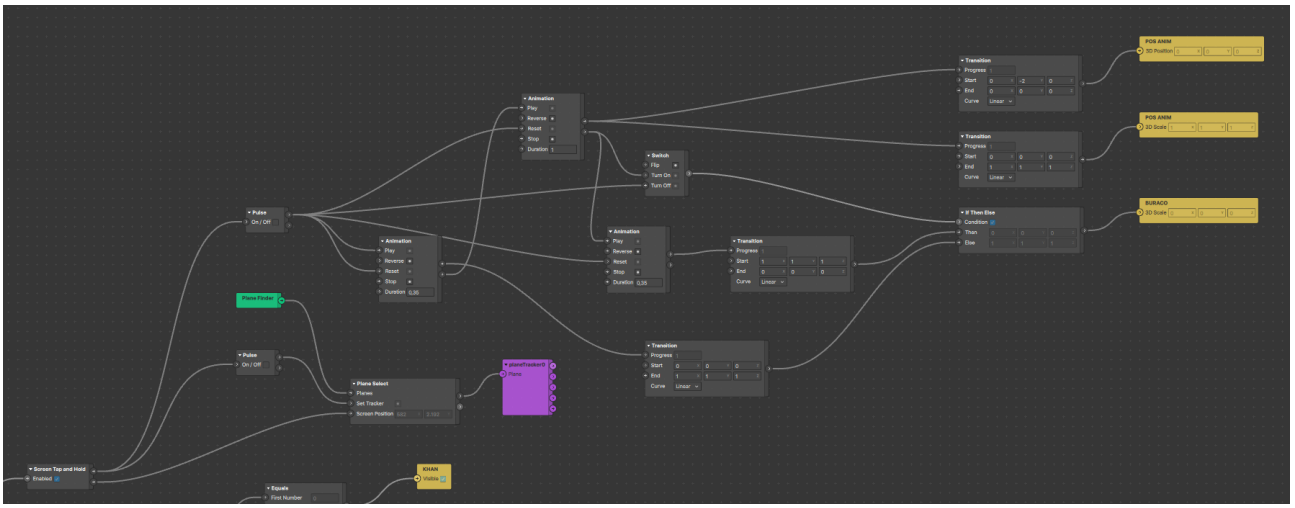
Dia 15 - Projeto #06. Khan Academy 3D AR Portal





©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri



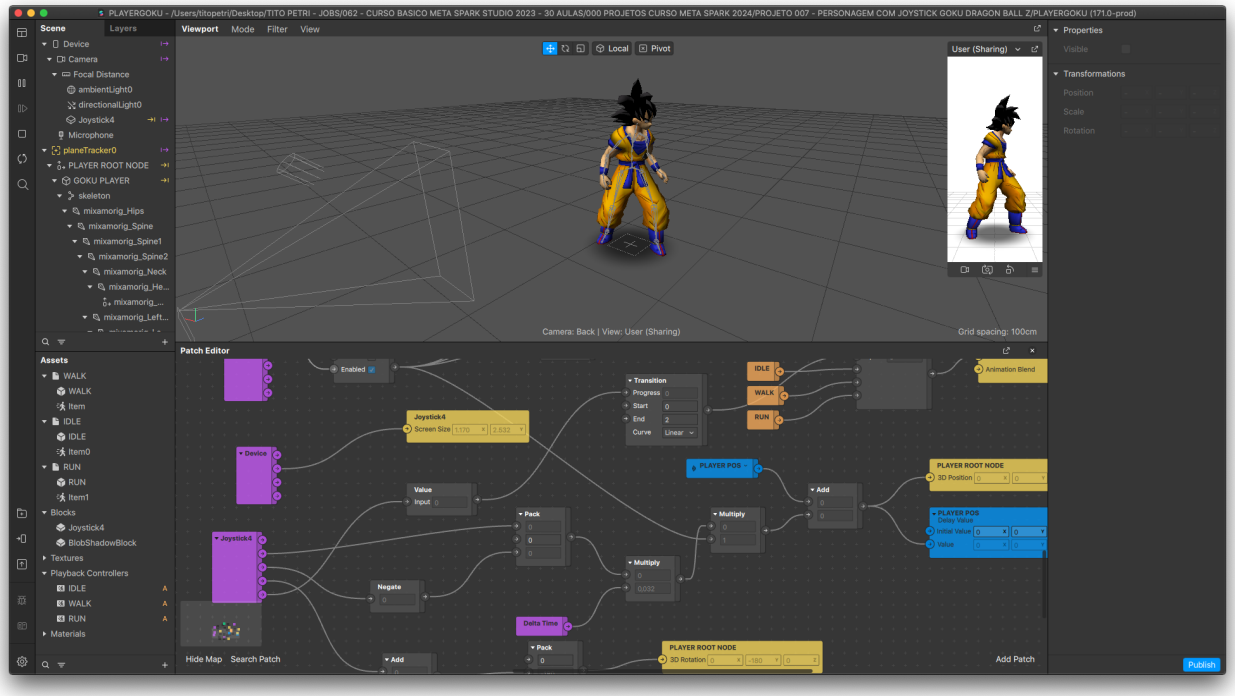


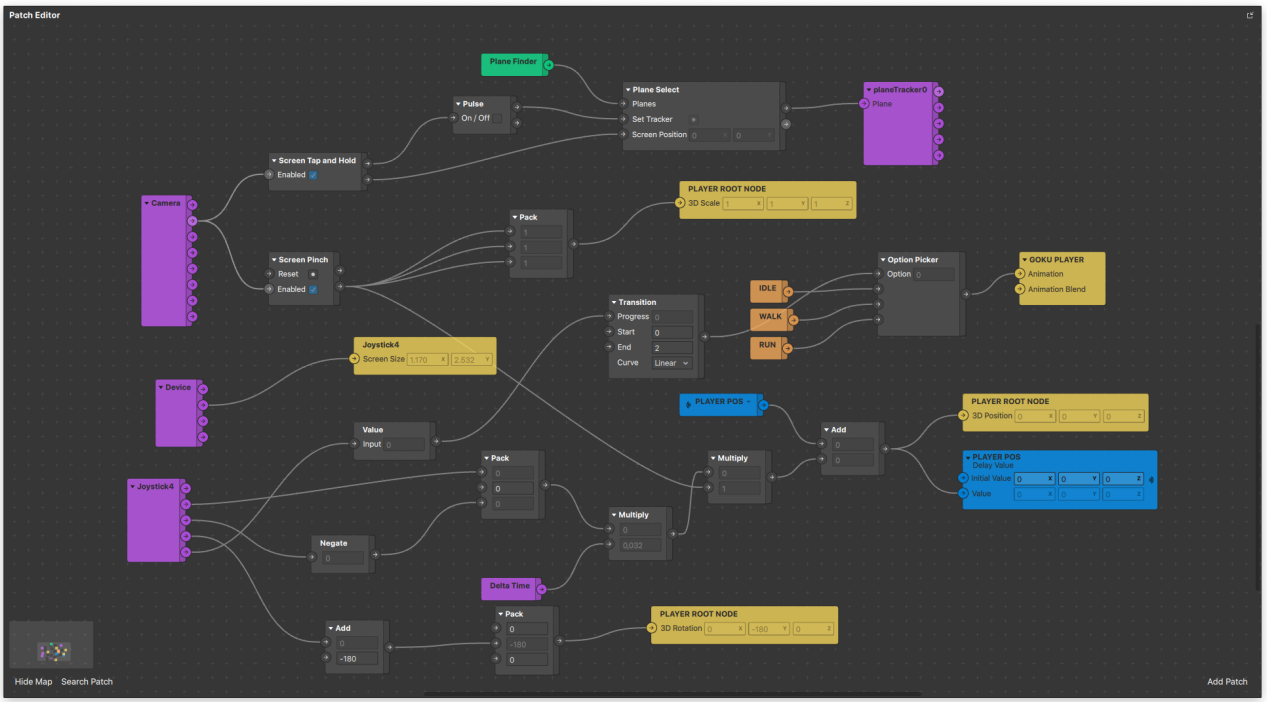
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

```
const Scene = require('Scene');
const HandTracking = require('HandTracking');
export const Diagnostics = require('Diagnostics');

let notHand;

( async function () {
  const nodeHand = await Scene.root.findFirst("nodeHand");
  const nodePlaneTracker = await Scene.root.findFirst("nodePlaneTracker");
  notHand = HandTracking.count.eq(0);
  const rootNodeLogo = await Scene.root.findFirst("LOGOS 3D ROOT NODE");

  notHand.onOn({fireOnInitialValue: true}).subscribe((event) => {
      if((nodePlaneTracker!=null)&&(rootNodeLogo!=null)){
        rootNodeLogo.removeFromParent();
        nodePlaneTracker.addChild(rootNodeLogo);
      }
    });
  notHand.onOff({fireOnInitialValue: true}).subscribe((event) => {

      if(rootNodeLogo!=null){
        rootNodeLogo.removeFromParent();
        nodeHand.addChild(rootNodeLogo);
      }
    });
})();
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

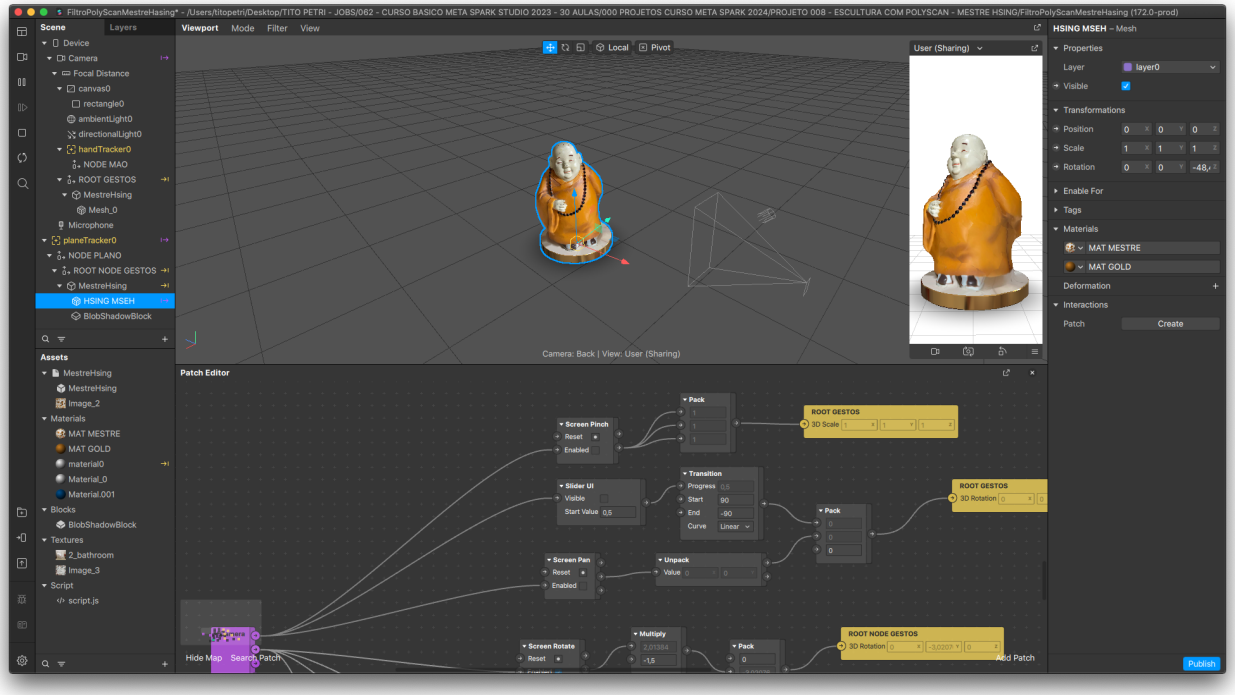
Dia 16 - Projeto #07. Personagem com Joystick Goku Dragon Ball-Z

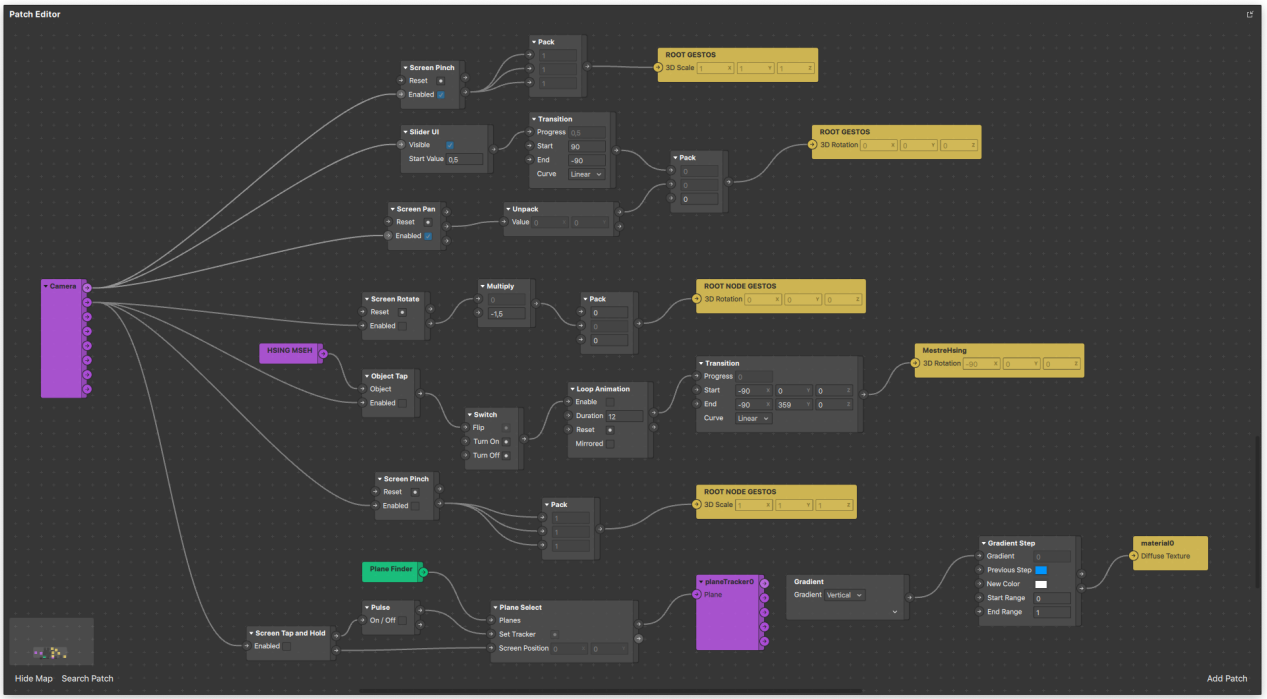




©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

Dia 17 - Projeto #08. Escultura com PolyScan Mestre Hsing Yün



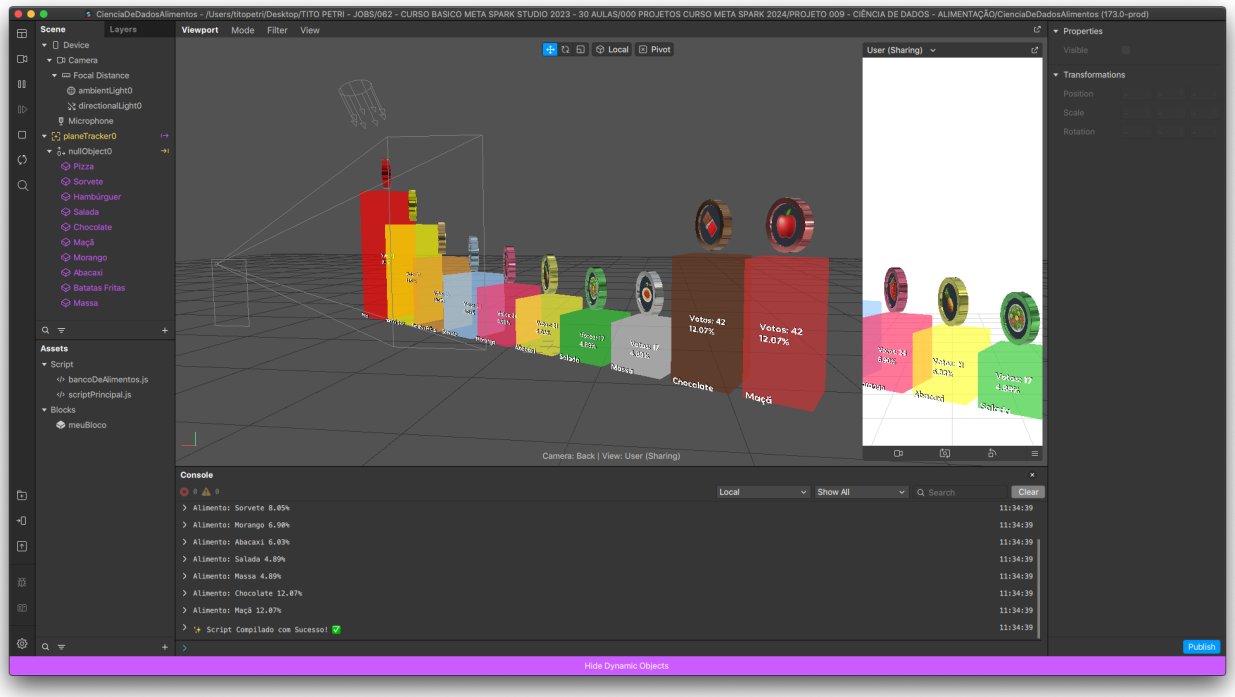


©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

```
const Scene = require('Scene');
const HandTracking = require('HandTracking');

export const Diagnostics = require('Diagnostics');

let notHand;

( async function () {

  const nodeHand = await Scene.root.findFirst("NODE MAO");
  const nodePlaneTracker = await Scene.root.findFirst("NODE PLANO");
  const rootNodeLogo = await Scene.root.findFirst("ROOT NODE GESTOS");

  notHand = HandTracking.count.eq(0);

  notHand.onOn({fireOnInitialValue: true}).subscribe((event) => {
      if((nodePlaneTracker!=null)&&(rootNodeLogo!=null)){
        rootNodeLogo.removeFromParent();
        nodePlaneTracker.addChild(rootNodeLogo);
      }
    });
  notHand.onOff({fireOnInitialValue: true}).subscribe((event) => {

      if(rootNodeLogo!=null){
        rootNodeLogo.removeFromParent();
        nodeHand.addChild(rootNodeLogo);
      }
    });
})();
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

Dia 18 - Projeto #09. Ciência de Dados Alimentação




```
import { meuBancoAlimentos } from './bancoDeAlimentos';

const Scene = require('Scene');
const Blocks = require('Blocks');
const Reactive = require('Reactive');

export const Diagnostics = require('Diagnostics');

(async function(){
  const rootCena = await Scene.root.findFirst("nullObject0");

  if(rootCena != null){
    for(let i=0; i<meuBancoAlimentos.length;i++){
      Blocks.instantiate('meuBloco',{
        name : meuBancoAlimentos[i].alimento,
        hidden : false
      }).then(function(bloco){
        bloco.transform.x = i * 0.05 + (i*0.01) - ((meuBancoAlimentos.length*0.06)/2);

        let indiceEmoji = 99;

        switch(meuBancoAlimentos[i].imagemUrl){
          case "imagemPizza": indiceEmoji = 0; break;
          case "imagemHamburguer": indiceEmoji = 1; break;
          case "imagemBatatasFritas": indiceEmoji = 2; break;
          case "imagemSorvete": indiceEmoji = 3; break;
          case "imagemMorango": indiceEmoji = 4; break;
          case "imagemAbacaxi": indiceEmoji = 5; break;
          case "imagemSalada": indiceEmoji = 6; break;
          case "imagemMassa": indiceEmoji = 7; break;
          case "imagemChocolate": indiceEmoji = 8; break;
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

```
          case "imagemMaca": indiceEmoji = 9; break;
          default: break;
        }
        bloco.inputs.setScalar('imagem',indiceEmoji);
        bloco.inputs.setScalar('votos', meuBancoAlimentos[i].votos*0.05);
        bloco.inputs.setString('nome', meuBancoAlimentos[i].alimento);

        const p = calculaPct(totalVotos, meuBancoAlimentos[i].votos);
        bloco.inputs.setString('informacoes', "Votos: "+meuBancoAlimentos[i].votos + "\n" + p.toFixed(2) +"%");

        var valorCorRGBA =
Reactive.RGBA(meuBancoAlimentos[i].cor[0],meuBancoAlimentos[i].cor[1],meuBancoAlimentos[i].cor[2],1.0);
        bloco.inputs.setColor('cor', valorCorRGBA);
        bloco.inputs.setPulse('iniciaAnimacao', Reactive.once());
        rootCena.addChild(bloco);
      });
    }
  }

})();



function numeroTotalDeVotos(){
  let totalVotos = 0;
  for(let i=0; i<meuBancoAlimentos.length;i++){
    totalVotos += meuBancoAlimentos[i].votos;
  }
  return totalVotos;
}
const totalVotos = numeroTotalDeVotos();

function calculaPct(numTotal,meuItem){
  return (meuItem/numTotal)*100;
}

function exibeTodosPercentuaisDePreferencia(){
  for(let i=0; i<meuBancoAlimentos.length;i++){
    const p = calculaPct(totalVotos, meuBancoAlimentos[i].votos);
    Diagnostics.log("Alimento: " + meuBancoAlimentos[i].alimento + " " + p.toFixed(2) +"%")
  }
}
exibeTodosPercentuaisDePreferencia();

Diagnostics.log("✨ Script Compilado com Sucesso! ✅");
```

```
export const meuBancoAlimentos = [
  {
    alimento: "Pizza",
    emoji: "🍕",
    imagemUrl: "imagemPizza",
    votos: 70,
    calorias: 285,
    proteinas: 12,
    carboidratos: 36,
    acucares: 3,
    cor: [1.0, 0.0, 0.0]
  },
  {
    alimento: "Hambúrguer",
    emoji: "🍔",
    imagemUrl: "imagemHamburguer",
    votos: 52,
    calorias: 250,
    proteinas: 15,
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

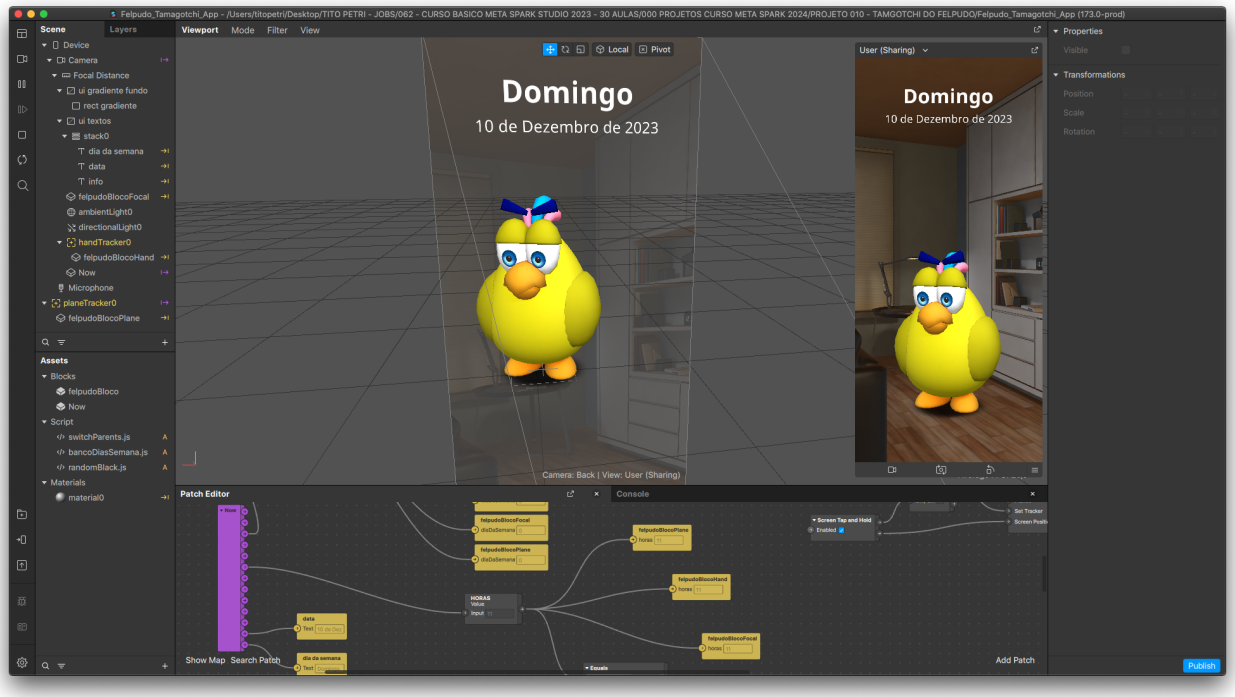
```
  carboidratos: 10,
  acucares: 2,
  cor: [1.0, 1.0, 0.0]
},
{
  alimento: "Batatas Fritas",
  emoji: "🍟",
  imagemUrl: "imagemBatatasFritas",
  votos: 35,
  calorias: 365,
  proteinas: 3,
  carboidratos: 63,
  acucares: 0.6,
  cor: [0.9, 0.6, 0.2]
},
{
  alimento: "Sorvete",
  emoji: "🍦",
  imagemUrl: "imagemSorvete",
  votos: 28,
  calorias: 137,
  proteinas: 2,
  carboidratos: 25,
  acucares: 24,
  cor: [0.6, 0.8, 1.0]
},
{
  alimento: "Morango",
  emoji: "🍓",
  imagemUrl: "imagemMorango",
  votos: 24,
  calorias: 32,
  proteinas: 0.7,
  carboidratos: 8,
  acucares: 4.9,
  cor: [1.0, 0.2, 0.4]
},
{
  alimento: "Abacaxi",
  emoji: "🍍",
  imagemUrl: "imagemAbacaxi",
  votos: 21,
  calorias: 50,
  proteinas: 0.5,
  carboidratos: 13,
  acucares: 9.9,
  cor: [1.0, 1.0, 0.2]
},
{
  alimento: "Salada",
  emoji: "🥗",
  imagemUrl: "imagemSalada",
  votos: 17,
  calorias: 15,
  proteinas: 1,
  carboidratos: 3,
  acucares: 1.5,
  cor: [0.2, 0.8, 0.2]
},
{
  alimento: "Massa",
  emoji: "🍝",
  imagemUrl: "imagemMassa",
  votos: 17,
  calorias: 200,
  proteinas: 7,
  carboidratos: 42,
  acucares: 2,
```

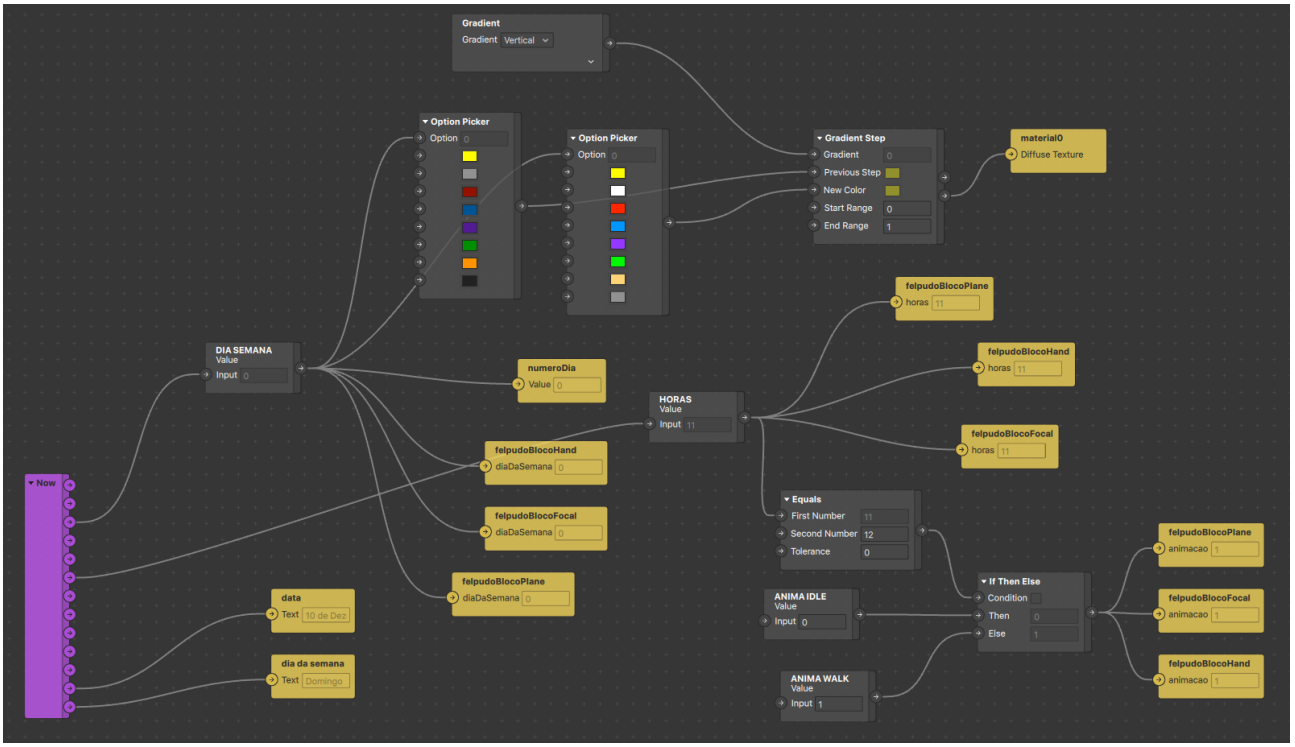
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

```
    cor: [0.8, 0.8, 0.8]
  },
  {
    alimento: "Chocolate",
    emoji: "🍫",
    imagemUrl: "imagemChocolate",
    votos: 42,
    calorias: 546,
    proteinas: 6,
    carboidratos: 66,
    acucares: 56,
    cor: [0.4, 0.2, 0.1]
  },
  {
    alimento: "Maçã",
    emoji: "🍎",
    imagemUrl: "imagemMaca",
    votos: 42,
    calorias: 52,
    proteinas: 0.3,
    carboidratos: 14,
    acucares: 10,
    cor: [0.8, 0.2, 0.2]
  }
];
```

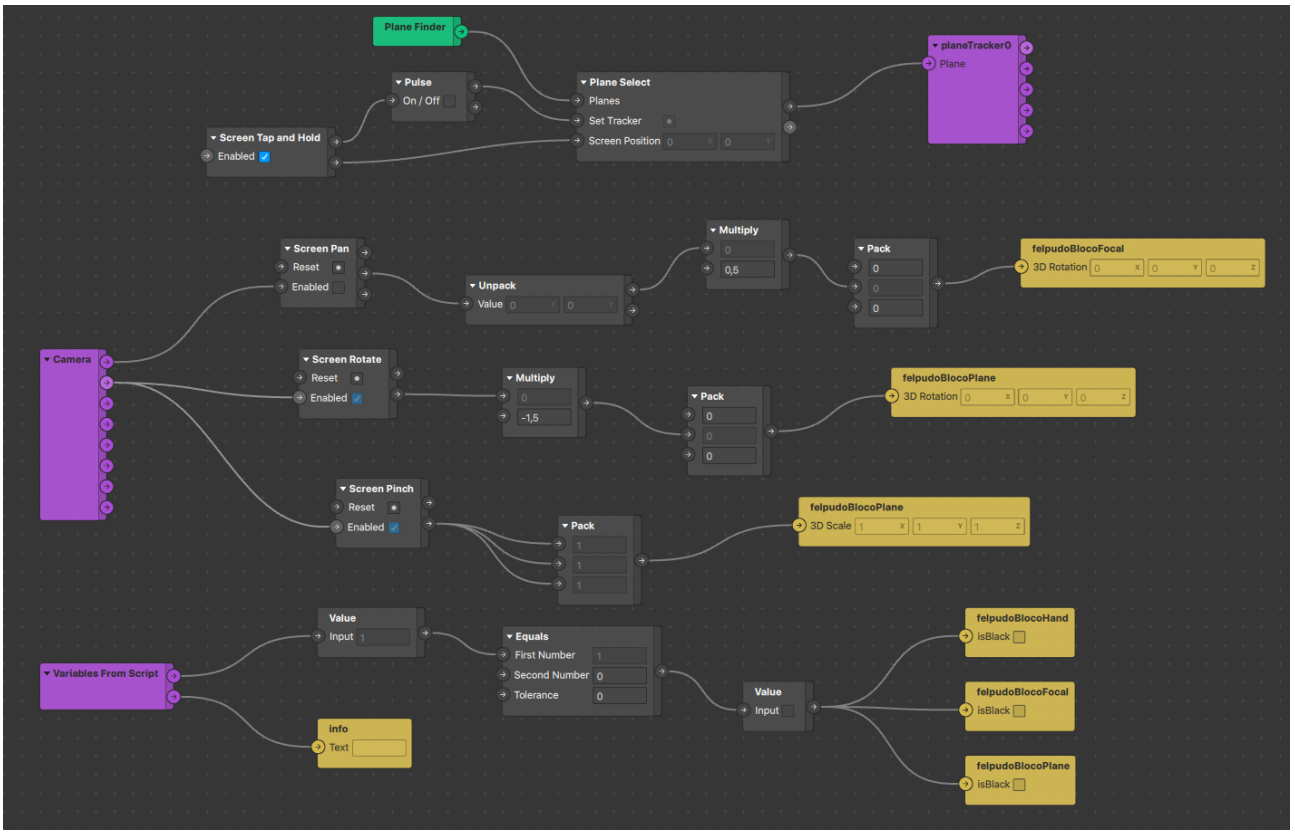
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

Dia 19 - Projeto #10. Tamagotchi do Felpudo





©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri




```
export const Diagnostics = require('Diagnostics');
const Scene = require('Scene');
const HandTracking = require('HandTracking');
let notHand;

( async function () {

  const nodeHand = await Scene.root.findFirst("handTracker0");
  //const nodePlaneTracker = await Scene.root.findFirst("planeTracker0");
  const rootNodePlayerH = await Scene.root.findFirst("felpudoBlocoHand");
  const rootNodePlayerP = await Scene.root.findFirst("felpudoBlocoPlane");
  notHand = HandTracking.count.eq(0);


  notHand.onOff({fireOnInitialValue: true}).subscribe((event) => {
    if((rootNodePlayerH!=null)&&(rootNodePlayerP!=null)){
      // rootNodePlayer.removeFromParent();
      // nodeHand.addChild(rootNodePlayer);
      rootNodePlayerP.hidden=true;
      rootNodePlayerH.hidden=false;
    }
  });

  notHand.onOn({fireOnInitialValue: true}).subscribe((event) => {
    if((rootNodePlayerH!=null)&&(rootNodePlayerP!=null)){
      // rootNodePlayer.removeFromParent();
      // nodePlaneTracker.addChild(rootNodePlayer);
      rootNodePlayerP.hidden=false;
      rootNodePlayerH.hidden=true;
    }
  });
})();



import { Diagnostics } from './switchParents';
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

```
const Patches = require('Patches');
let textoFormatado = "texto infos";

export const meuBanco = [
 {
  dia: "Domingo",
  diaIngles: "Sunday",
  astro: "Sol",
  emojiAstro: "☀️",
  cor: "Amarelo ou Dourado",
  emojiCor: "💛",
  nomeChackra: "Chakra do Plexo Solar",
  nomeChackraSanscrito: "Manipura"
 },{
  dia: "Segunda-feira",
  diaIngles: "Monday",
  astro: "Lua",
  emojiAstro: "🌙",
  cor: "Prateado ou Branco",
  emojiCor: "🌕",
  nomeChackra: "Chakra da Raiz",
  nomeChackraSanscrito: "Muladhara"
 },
 {
  dia: "Terça-feira",
  diaIngles: "Tuesday",
  astro: "Marte",
  emojiAstro: "♂",
  cor: "Vermelho",
  emojiCor: "❤️",
  nomeChackra: "Chakra Sacral",
  nomeChackraSanscrito: "Svadhisthana"
 },
 {
  dia: "Quarta-feira",
  diaIngles: "Wednesday",
  astro: "Mercúrio",
  emojiAstro: "☿",
  cor: "Azul ou Cinza",
  emojiCor: "🔵",
  nomeChackra: "Chakra da Garganta",
  nomeChackraSanscrito: "Vishuddha"
 },
 {
  dia: "Quinta-feira",
  diaIngles: "Thursday",
  astro: "Júpiter",
  emojiAstro: "♃",
  cor: "Roxo ou Azul-escuro",
  emojiCor: "💜",
  nomeChackra: "Chakra do Terceiro Olho",
  nomeChackraSanscrito: "Ajna"
 },
 {
  dia: "Sexta-feira",
  diaIngles: "Friday",
  astro: "Vênus",
  emojiAstro: "♀",
  cor: "Verde ou Rosa",
  emojiCor: "💚",
  nomeChackra: "Chakra Cardíaco",
  nomeChackraSanscrito: "Anahata"
 },
 {
  dia: "Sábado",
  diaIngles: "Saturday",
  astro: "Saturno",
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

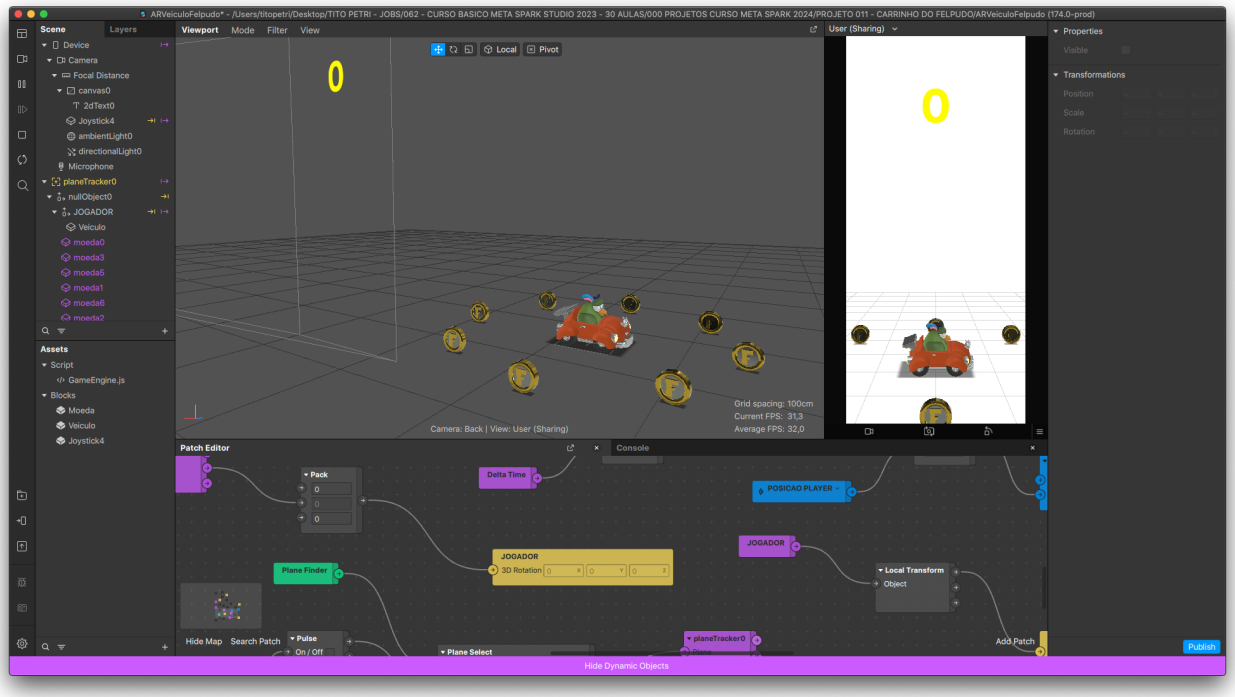
```
    emojiAstro: "♑",
    cor: "Preto ou Marrom",
    emojiCor: "⚫",
    nomeChackra: "Chakra da Coroa",
    nomeChackraSanscrito: "Sahasrara"
  }
];

(async function(){
  const numeroDia = await Patches.outputs.getScalar('numeroDia').then(event=> {
    event.monitor().subscribe(function (values) {
        textoFormatado = meuBanco[values.newValue].diaIngles;
        textoFormatado += "\n " + meuBanco[values.newValue].astro;
        textoFormatado += " - " + meuBanco[values.newValue].cor;
        textoFormatado += "\n " + meuBanco[values.newValue].nomeChackra;
        textoFormatado += " (" + meuBanco[values.newValue].nomeChackraSanscrito + ")";
        Patches.inputs.setString('textoInfoDia', textoFormatado);
    });
});
})();


// const myBoolean = true;
// const myString = await Patches.outputs.getString('myString');
// const myString = await Patches.outputs.getString('myString');
// await Patches.inputs.setBoolean('myBoolean', myBoolean);
// const randomInt = Math.floor(Math.random()*101);
// Diagnostics.log(randomInt);

export const Diagnostics = require('Diagnostics');
const Patches = require('Patches');

( async function(){
  const randomNumeroBlack = Math.floor(Math.random()*2);
  await Patches.inputs.setScalar('randomBlack', randomNumeroBlack);
})();
Diagnostics.log("✅ Script Compilado com Sucesso! ✨");


const Patches = require('Patches');
const Time = require('Time');
export const Diagnostics = require('Diagnostics');

let _keepUpdate = true;

let nomeMes;
let horaFormatada;
let diaDaSemana;

(async function () {

  function update(){
    let now = new Date(Date.now());

    switch(now.getDay()){
        case 0: diaDaSemana = "Domingo"; break;
        case 1: diaDaSemana = "Segunda-Feira"; break;
        case 2: diaDaSemana = "Terça-Feira"; break;
        case 3: diaDaSemana = "Quarta-Feira"; break;
        case 4: diaDaSemana = "Quinta-Feira"; break;
        case 5: diaDaSemana = "Sexta-Feira"; break;
        case 6: diaDaSemana = "Sábado"; break;
        default: break;
    }
    Patches.inputs.setString('diaDaSemana',diaDaSemana);

    Patches.inputs.setScalar('month',now.getMonth());
    // Diagnostics.log(now.getMonth());
```

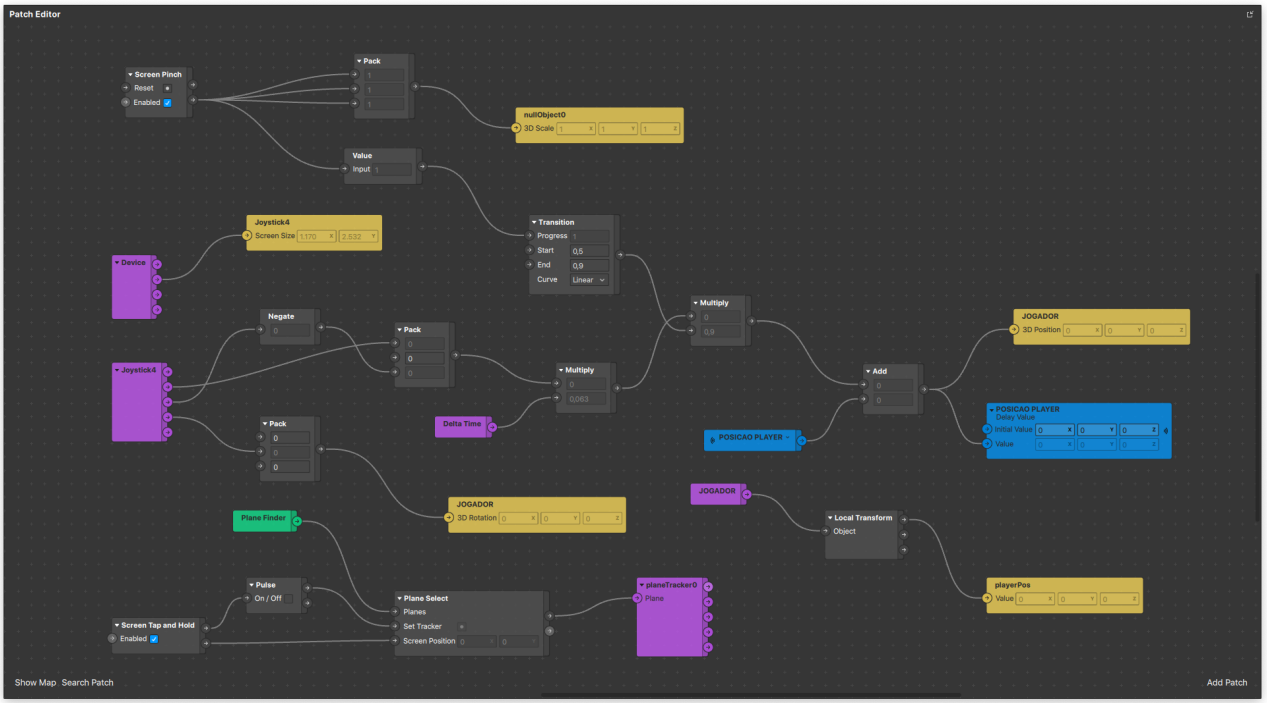
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

```
    switch(now.getMonth()){
      case 0: nomeMes = "Janeiro"; break;
      case 1: nomeMes = "Fevereiro"; break;
      case 2: nomeMes = "Março";  break;
      case 3: nomeMes = "Abril";  break;
      case 4: nomeMes = "Maio";  break;
      case 5: nomeMes = "Junho";  break;
      case 6: nomeMes = "Julho";  break;
      case 7: nomeMes = "Agosto";  break;
      case 8: nomeMes = "Setembro";  break;
      case 9: nomeMes = "Outubro";  break;
      case 10: nomeMes = "Novembro";  break;
      case 11: nomeMes = "Dezembro";  break;
    }
    //Diagnostics.log(nomeMes);

    const str = ""+now.getDate()+" de "+nomeMes+" de "+now.getFullYear();
    Patches.inputs.setString('dataFormatada',str);

    Patches.inputs.setScalar('date',now.getDate());
    Patches.inputs.setScalar('day',now.getDay());
    Patches.inputs.setScalar('fullyear',now.getFullYear());
    Patches.inputs.setScalar('hours',now.getHours());
    Patches.inputs.setScalar('minutes',now.getMinutes());
    Patches.inputs.setScalar('seconds',now.getSeconds());
    Patches.inputs.setScalar('milliseconds',now.getMilliseconds());
    Patches.inputs.setScalar('time',now.getTime());
    Patches.inputs.setScalar('timezoneoffset',now.getTimezoneOffset());

  }

  const keepUpdate = await Patches.outputs.getBoolean('keepUpdate');
  const updateNow = await Patches.outputs.getPulse('updateNow');

  keepUpdate.monitor().subscribe(({newValue})=>{
    _keepUpdate = newValue;
  });

  updateNow.subscribe(update);

  Time.ms.monitor().subscribe(()=>{
    if(_keepUpdate){
      update();
    }
  });
  update();

})();
Diagnostics.log("✅ Script Compilado com Sucesso! ✨");
```

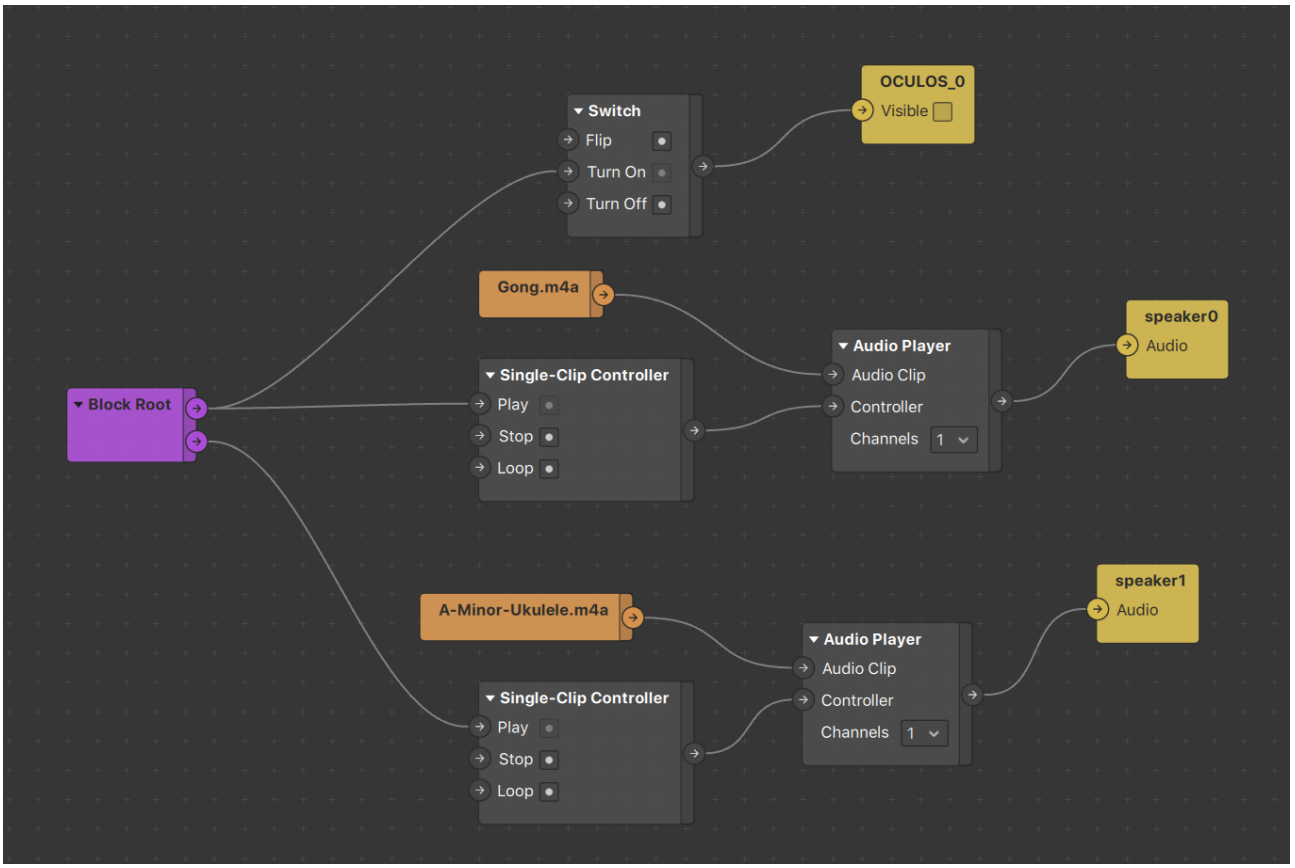
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

Dia 20 - Projeto #11. Veículo Do Felpudo





©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri



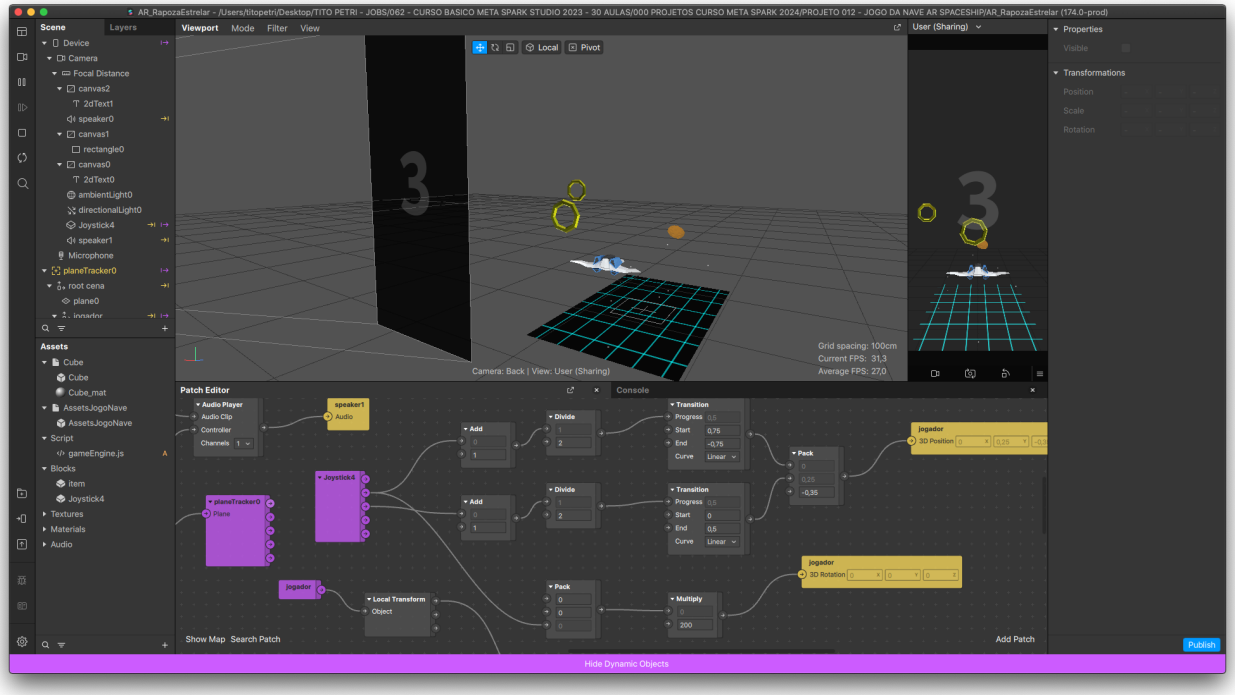

```
const Diagnostics = require('Diagnostics');
const Scene = require('Scene');
const Blocks = require('Blocks');
const Patches = require('Patches');
const Reactive = require('Reactive');
const TouchGestures = require('TouchGestures');

export let listaItens=[];

let _score = 0;
var pulsed;
let _txtScore;
let quantidadeItens = 8;
const raio = 0.5;
const anguloIncremental = (2 * Math.PI) / quantidadeItens;
let blocoCarro;

(async function(){

  const rootNode = await Scene.root.findFirst('nullObject0');
  const posicaoJogador = await Patches.outputs.getVector('playerPos');

  blocoCarro = await Scene.root.findFirst('Veiculo');
  _txtScore = await Scene.root.findFirst("2dText0");

  for(let i=0;i<quantidadeItens;i++){
    Blocks.instantiate('Moeda',{
      name : 'moeda'+i,
      hidden : false
    }).then(function(meuBloco){

      // meuBloco.transform.x = 0.1*i;
      // meuBloco.transform.z = 0.1*i;
      // meuBloco.transform.y = 0.0;
```

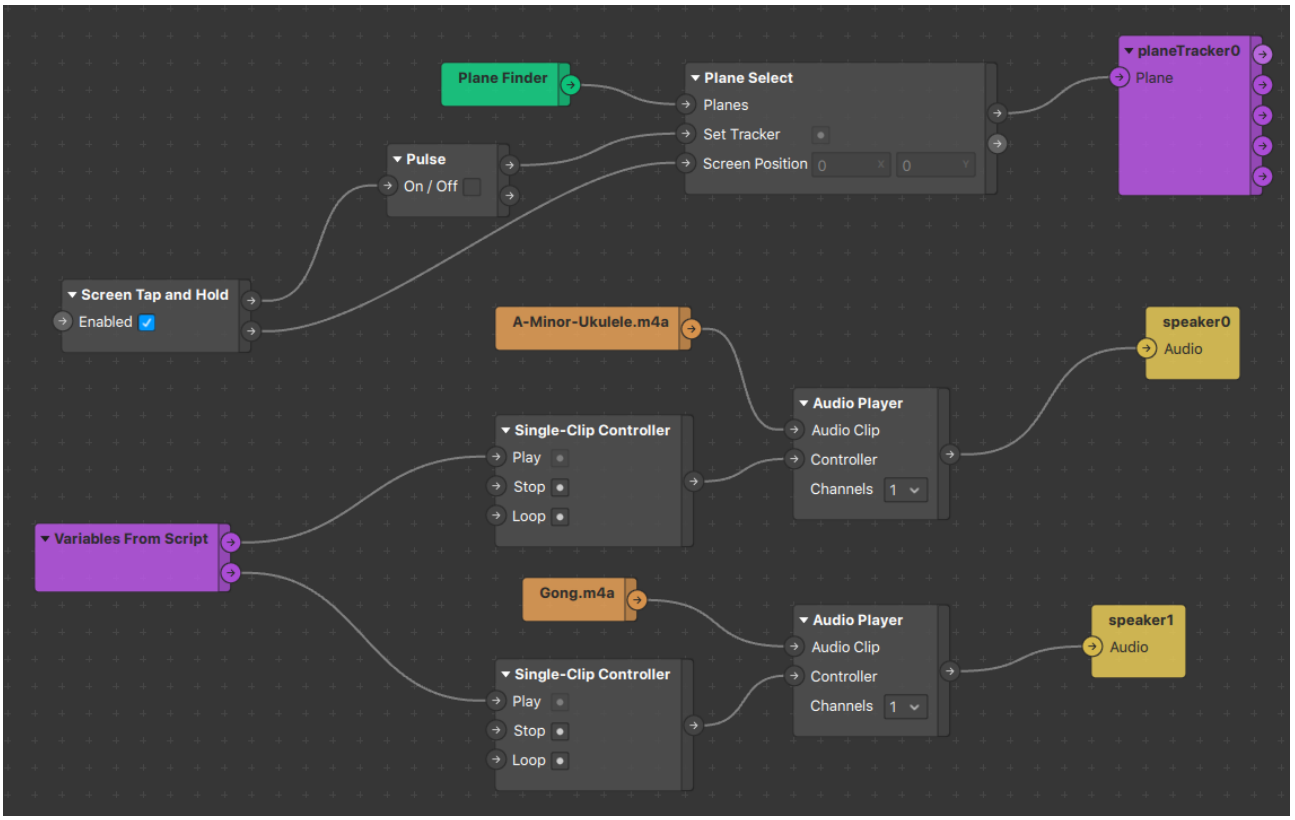
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

```
      // const _vector = Reactive.vector(meuBloco.transform.x, meuBloco.transform.y, meuBloco.transform.z);

      const angulo = i * anguloIncremental;

      const _x = raio * Math.cos(angulo);
      const _z = raio * Math.sin(angulo);

      meuBloco.transform.y = 0.0;
      meuBloco.transform.x = _x;
      meuBloco.transform.z = _z;
      const _vector = Reactive.vector(_x, 0.0, _z);

      meuBloco.inputs.setScalar('indice', i);
      meuBloco.inputs.setVector('playerPos', posicaoJogador);
      meuBloco.inputs.setVector('itemPos', _vector);

      listaItens[i] = meuBloco;

      rootNode.addChild(meuBloco).then(event =>{
        if(i==quantidadeItens-1){
          loopListaItens();
        }
      });
    });
  }
})();

export function apagaMoeda(indiceMoeda){
  Scene.destroy(listaItens[indiceMoeda]);
}
function atualizaUITexto(){
  _txtScore.text = ""+_score;
  blocoCarro.inputs.setPulse('pulsoMoeda', Reactive.once());
  if(_score==quantidadeItens){
    _txtScore.text="Felpudo Wins!";
    felpudoWins();
  }
}

function felpudoWins(){
  Diagnostics.log(blocoCarro.name);
  blocoCarro.inputs.setPulse('pulsoFelpudoWins', Reactive.once());
}

function loopListaItens(){
  _txtScore.text = ""+_score;

  for(let i=0;i<listaItens.length;i++){
    listaItens[i].outputs.getPulse('pulsoDestroiItem').then(event => {
        pulsed = event.subscribe(function () {
            apagaMoeda(i);
            // _score++;
            // atualizaUITexto();
        });
    });

    listaItens[i].outputs.getPulse('playSound').then(event => {
      pulsed = event.subscribe(function () {
          // apagaMoeda(i);
          _score++;
          atualizaUITexto();
      });
    });

  }
}
```

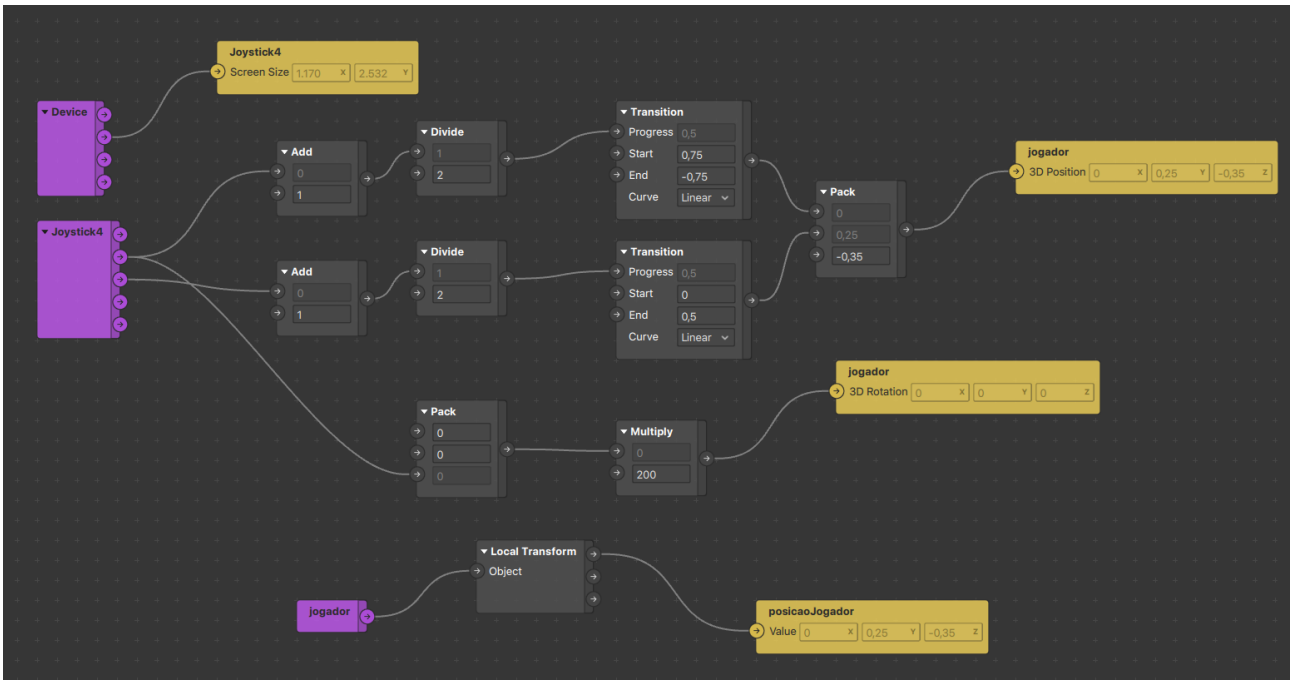
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

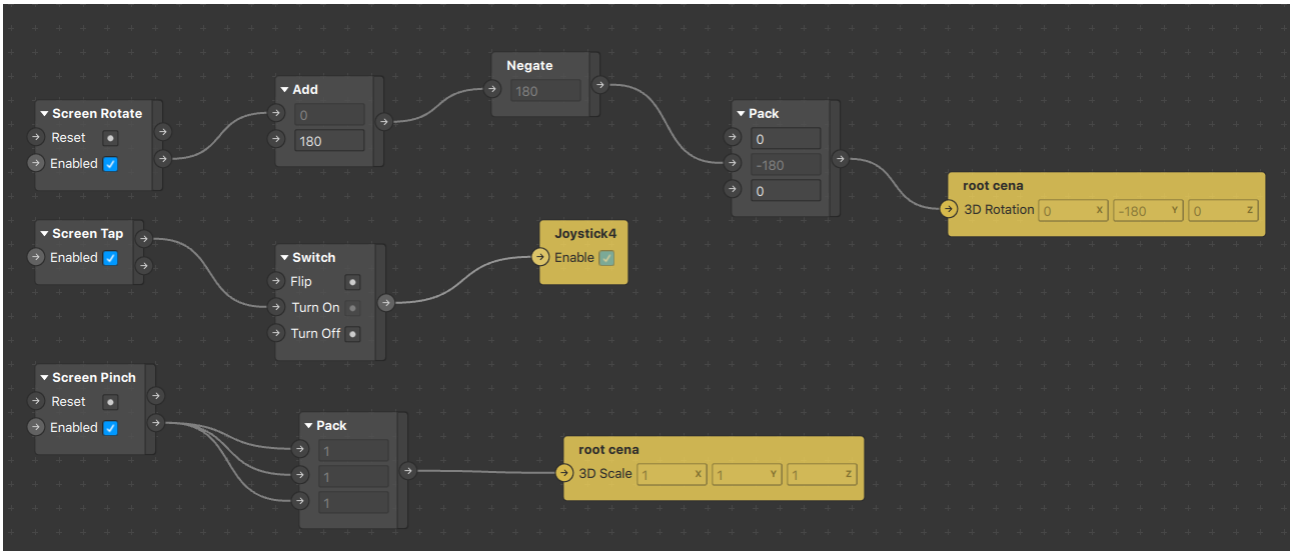
Dia 21 - Projeto #12. AR Raposa Estelar

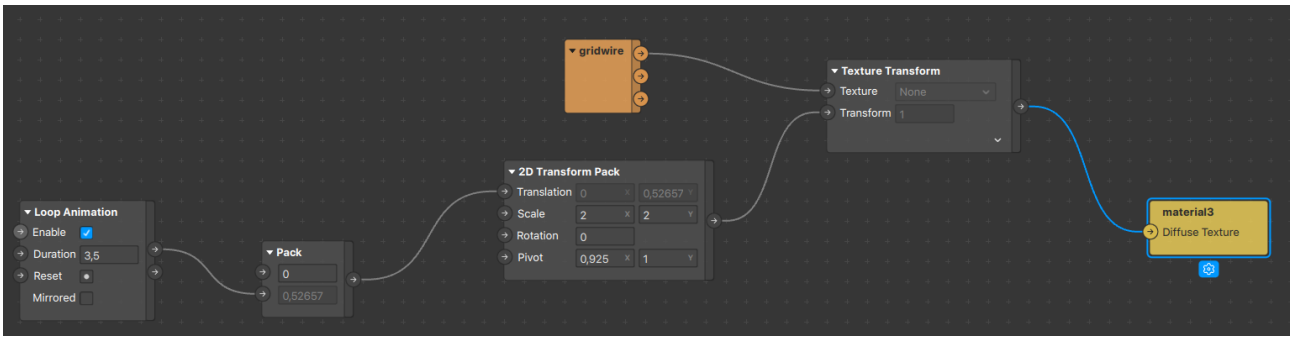




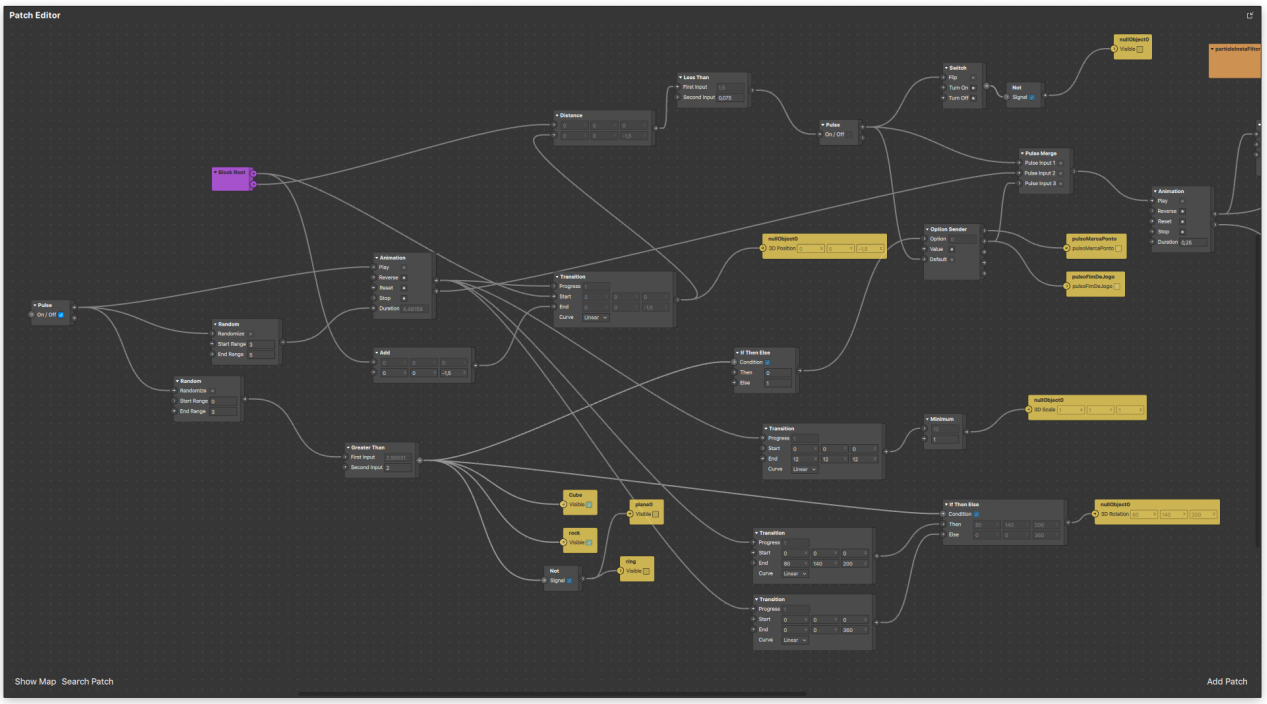
©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

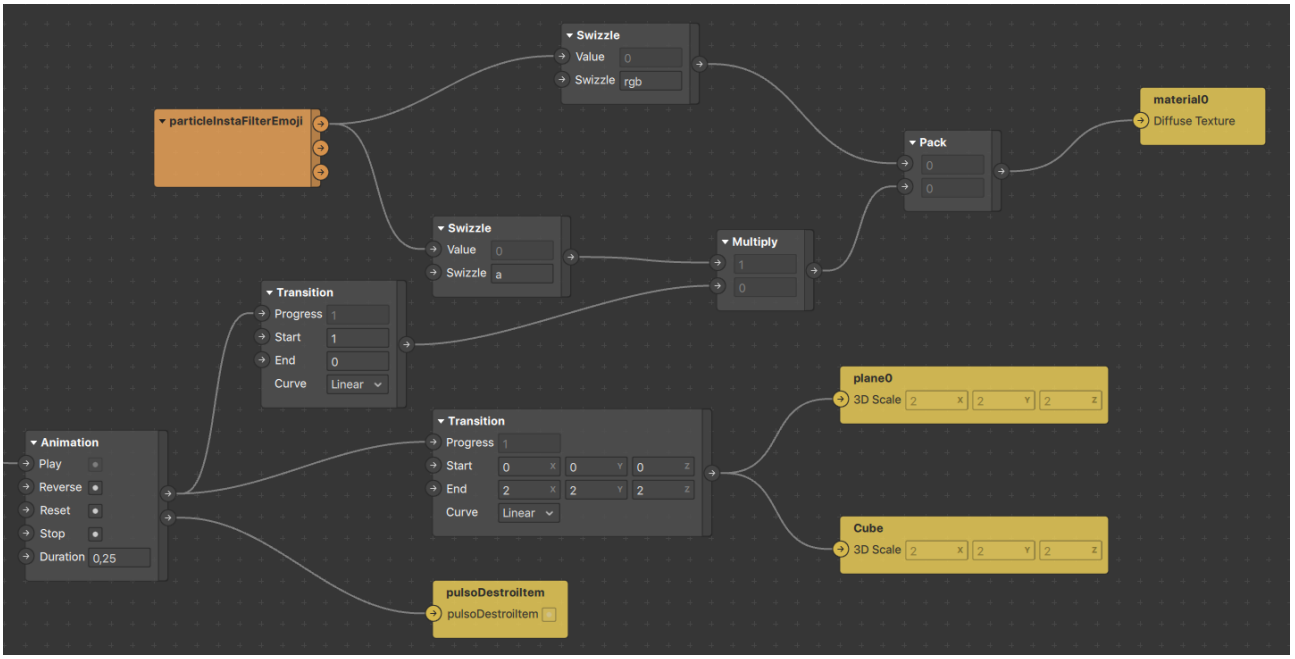






©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri






```
const Diagnostics = require('Diagnostics');
const Patches = require('Patches');
const Reactive = require('Reactive');
const Blocks = require('Blocks');
const Scene = require('Scene');
const TouchGestures = require('TouchGestures');
const Time = require('Time');

let rootNode;
let _ringTime = 500;
let posicaoJogador;
let timerItens;
let textoScore;
let textoInstruct;
let _score = 0;
let gameStarted = false;
let gameOver = false;
let jogador;
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

```
let itemid=0;
let _mensagem = "Toque para iniciar";

async function marcaPonto(){
  if(!gameOver){
    _score++;
    Patches.inputs.setPulse('pulsoMarcaPonto', Reactive.once());
    textoScore.text = ""+_score;
  }
}

async function fimDeJogo(){
  if(!gameOver){
    gameOver=true;
    textoInstruct.text = _mensagem;
    textoInstruct.hidden=false;
    Patches.inputs.setPulse('pulsoLose', Reactive.once());
    Time.clearInterval(timerItens);
    gameOver = true;
    gameStarted = false;
    _score = 0;
    jogador.hidden = gameOver;
  }
}


async function spawnRing(){
  itemid++;
  Blocks.instantiate('item',{
    name : 'meuItem'+itemid,
    hidden: true
  }).then(function(bloco){
    var pX = (Math.random() * 0.5) - 0.25;
    var pY = Math.random() * 0.5;
    var pZ = 0.5;

    var pointSignal = Reactive.point(pX, pY, pZ);

    bloco.inputs.setPoint('randomPosition', pointSignal);
    bloco.inputs.setVector('posicaoJogador', posicaoJogador);

    rootNode.addChild(bloco).then(event => {
        bloco.outputs.getPulse('pulsoDestroiItem').then(event => {
          let pulsed = event.subscribe(function () {
            Scene.destroy(bloco);
          });
        });

        bloco.outputs.getPulse('pulsoMarcaPonto').then(event => {
          let pulsed = event.subscribe(function () {
            marcaPonto();
          });
        });

        bloco.outputs.getPulse('pulsoFimDeJogo').then(event => {
          let pulsed = event.subscribe(function () {
            fimDeJogo();

          });
        });
      bloco.hidden = false;
    });
  });
}



( async function (){
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri

```
    rootNode = await Scene.root.findFirst('root cena');
    textoInstruct = await Scene.root.findFirst('2dText0');
    textoScore = await Scene.root.findFirst('2dText1');

    jogador = await Scene.root.findFirst('jogador');

    posicaoJogador = await Patches.outputs.getVector('posicaoJogador');
    textoScore.text = ""+_score;
    textoInstruct.text = "Toque para iniciar";
    textoInstruct.hidden=false;

    TouchGestures.onTap().subscribe(() => {
      if(gameStarted == false){
        gameOver=false;
        jogador.hidden=gameOver;
        gameStarted = true;
        _ringTime = Math.random()*1000+1000;
        timerItens = Time.setInterval(spawnRing, _ringTime);
        textoInstruct.hidden=true;
        textoScore.hidden=false;
        textoScore.text = ""+_score;
      }
    });
})();
```

©Tito Petri - 2024 Todos os Direitos Reservados. @tito.petri